Solving Inverse Problems with Invertible Neural Networks

Ullrich Köthe

Visual Learning Lab, Heidelberg University

joint work with **Lynton Ardizzone, Stefan Radev, Jakob Kruse, Peter Sorrenson, Tim Adler, Sebastian Wirkert, Victor Ksoll, Anja Butter, Armand Rousselot, Tilman Plehn, Ralf Klessen, Carsten Rother, Lena Maier-Hein**

CERN IML workshop, October 2020

UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

Visual Learning Lab Heidelberg

European Research Council
Established by the European Commission

Generative Modelling

- Deep learning success story
  - Compute predictions y directly from complex data x
  - Point estimates: $\hat{y} \approx y^* = \operatorname{argmax} p(y \mid x)$, posteriors: $\hat{p}_\theta(y \mid x) \approx p(y \mid x)$
  - Relies on **discriminative / transductive machine learning**
    (does not first build a "model of the world" as traditional sciences do)
- Problem: discriminative models are hard to interpret, explain, validate

⇨ Generative modelling

- Turn the problem around: learn the data generation likelihood $p(x \mid y)$
- More difficult: requires ***insight*** beyond mere prediction capability
- Solve the original task via Bayes theorem

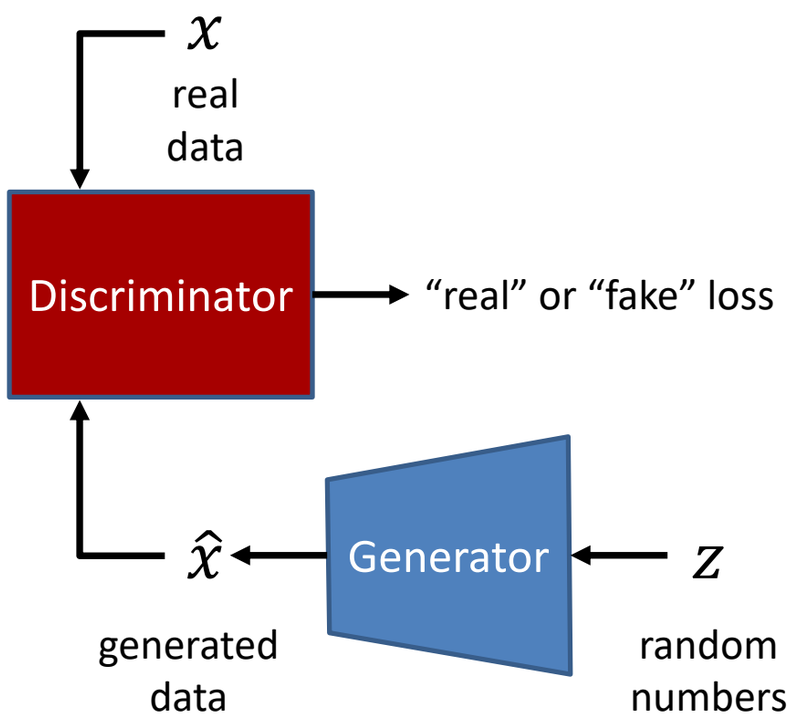
$$p(y \mid x) = \frac{p(x \mid y)\, p(y)}{p(x)}$$

Feynman: "What I cannot create, I do not understand."

Generative Modelling with Neural Networks

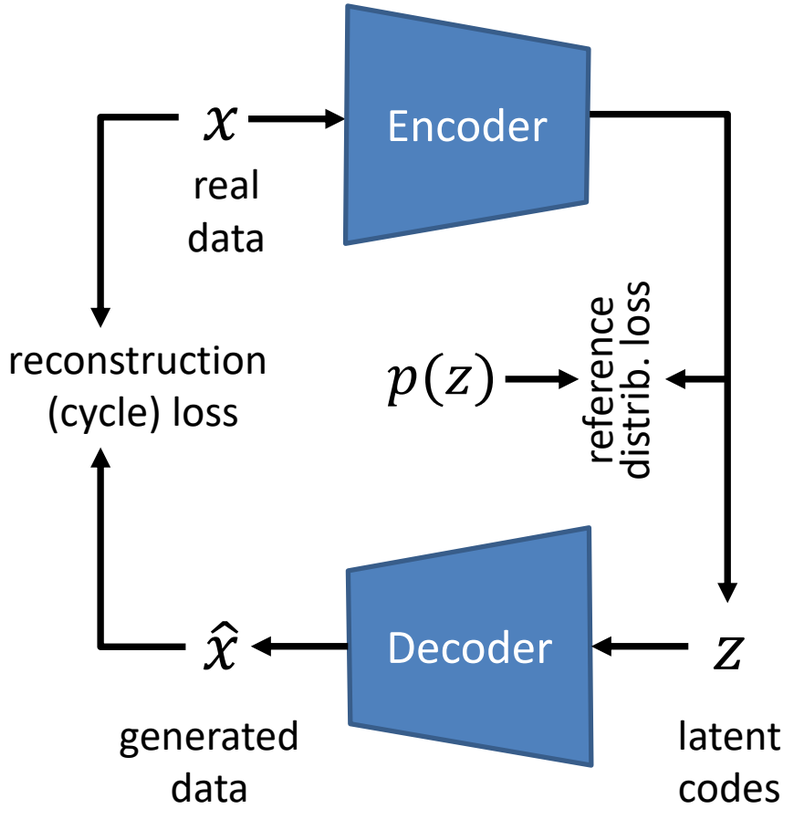
GANs

(Generative Adversarial Networks)



pure generation

(Variational) Autoencoders



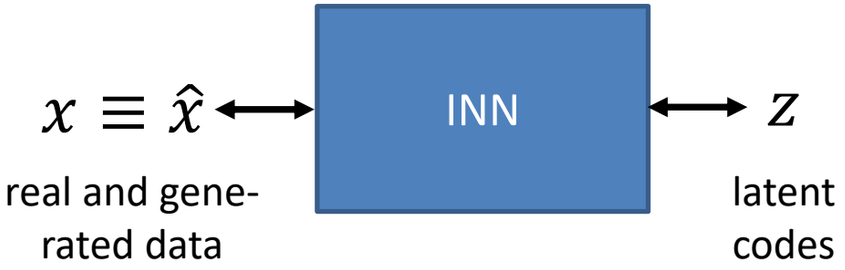
lossy encoding / decoding

Invertible Neural Networks

(INNs, normalizing flows)

maximum likelihood loss

$$p(x) = p\big(z = f(x)\big) \cdot |\det \nabla f|$$



Encoder and Decoder
the are same network,
run forward / backward

lossless encoding / decoding

Multiple Possibilities for INNs

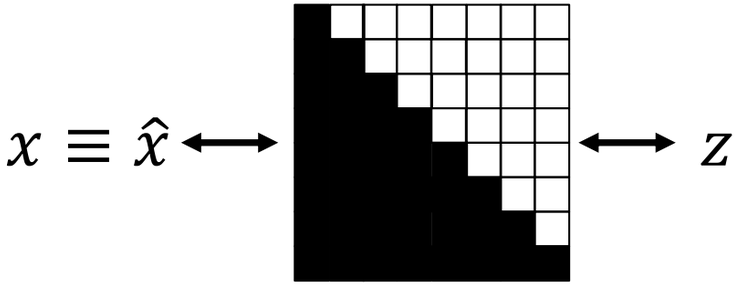
Autoregressive Models

Chain rule decomposition:

$$p(x_1, \dots, x_D) = \prod_i p_i(x_i \mid x_{<i})$$

triangular reparameterization:

$\forall i$: $x_i = f_i(z_i, x_{<i})$ monoton.



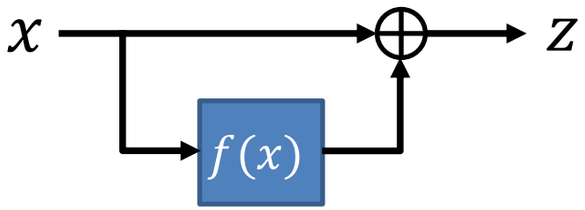
inverse direction inefficient

⇨ use two complementary nets

example: parallel WaveNet

iResNets
(invertible residual networks)

Residual block:



$$z = x + f(x)$$

is invertible when

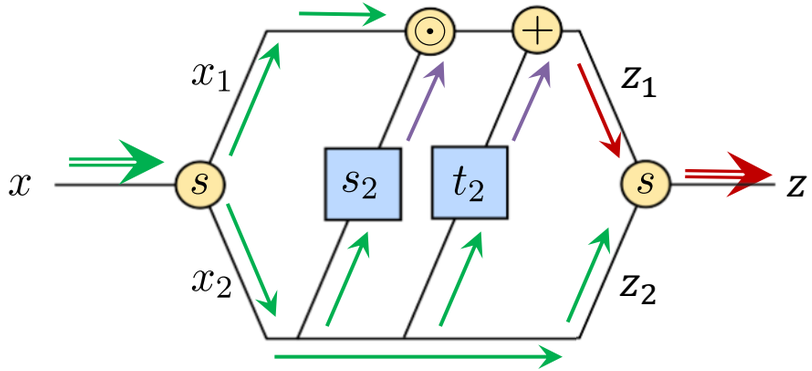
$$\|f(x)\|_{\text{Lipshitz}} < 1$$

inverse direction is reasonably efficient (fixpoint or Newton iterations)

example: Residual Flow Net

RealNVP

Affine coupling layer:



$$z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} x_1 \cdot s_2(x_2) + t_2(x_2) \\ x_2 \end{bmatrix}$$

inverse is equally efficient:

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} (z_1 - t_2(z_2))/s(z_2) \\ z_2 \end{bmatrix}$$

example: GLOW

Simulation-Based Inference

- **Inverse problem goal:** given observations $\hat{y}$, determine underlying hidden parameters $\hat{x}$
- In many inverse problems, the forward process is well understood…
  - Differential equations
  - Markov chains
  - Monte-Carlo simulations
  - …

$$\boldsymbol{y} = g(\boldsymbol{x}; \boldsymbol{\xi})$$

with $\boldsymbol{x}$ the hidden parameters and $\boldsymbol{\xi}$ a noise vector (i.e. the random numbers used in the simulation)

… but inversion is still difficult:

- likelihood $p(\boldsymbol{y} \mid \boldsymbol{x})$ of observed data $\boldsymbol{y}$ in Bayes formula

$$p(\boldsymbol{x} \mid \boldsymbol{y}) = \frac{p(\boldsymbol{y} \mid \boldsymbol{x})\, p(\boldsymbol{x})}{p(\boldsymbol{y})}$$

  is only implicitly defined by the simulation $\boldsymbol{y} = g(\boldsymbol{x}; \boldsymbol{\xi})$

⇨ **likelihood cannot be evaluated, posterior is analytically untractable**

“likelihood-free inference”

Simulation-Based Inference

- Classical simplification: reduce posterior to point estimate
  - **Regularization:** disambiguate inverse when forward process is surjective (not information preserving)
  - Maximum a-posteriori (**MAP**) inference: find only mode of posterior
  - ⇨ No idea of solution diversity and uncertainty
- Standard solution: approximate Bayesian computation (ABC)
  - Define a distance $d(\boldsymbol{y}_{\text{obs}}, \boldsymbol{y}_{\text{sim}})$ between observed and simulated data
    - for m=1,…, M:

      Sample hidden parameters $\boldsymbol{x}^{(m)} \sim p(\boldsymbol{x})$ from prior

      Run the simulation $\boldsymbol{y}_{\text{sim}}^{(m)} = g(\boldsymbol{x}^{(m)}; \boldsymbol{\xi})$

      Keep $\boldsymbol{x}^{(m)}$ if $d\left(\boldsymbol{y}_{\text{obs}}, \boldsymbol{y}_{\text{sim}}^{(\boldsymbol{m})}\right) < \epsilon$, reject otherwise
    - Return the set of "surviving" $\boldsymbol{x}^{(m)}$ as an approximate sample from $p(\boldsymbol{x} \mid \boldsymbol{y}_{\text{obs}})$
  - ⇨ Very slow: high rejection rate, because small ϵ are needed for accurate results
- Case-based inference
  - Efficient sampling methods (MCMC, SMC, …) with good proposal distribution have low rejection rates
  - ⇨ Learn a good proposal distribution for each observed data set $\boldsymbol{y}_{\text{obs}}$, i.e. on a per case basis
  - ⇨ Still expensive if many different $\boldsymbol{y}_{\text{obs}}$ must be evaluated

Simulation-Based Inference

- Classical simplification: reduce posterior to point estimate
  - **Regularization:** disambiguate inverse when forward process is surjective (not information preserving)
  - Maximum a-posteriori (**MAP**) inference: find only mode of posterior
  - ⇨ No idea of solution diversity and uncertainty
- Standard solution: approximate Bayesian computation (ABC)
  - Define a distance $d(\boldsymbol{y}_{\text{obs}}, \boldsymbol{y}_{\text{sim}})$ between observed and simulated data

We can do much better using invertible neural networks!

  - Return the set of "surviving" $\boldsymbol{x}^{(m)}$ as an approximate sample from $p(\boldsymbol{x} \mid \boldsymbol{y}_{\text{obs}})$
  - ⇨ Very slow: high rejection rate, because small ϵ are needed for accurate results
- Case-based inference
  - Efficient sampling methods (MCMC, SMC, …) with good proposal distribution have low rejection rates
  - ⇨ Learn a good proposal distribution for each observed data set $\boldsymbol{y}_{\text{obs}}$, i.e. on a per case basis
  - ⇨ Still expensive if many different $\boldsymbol{y}_{\text{obs}}$ must be evaluated

Linear Toy Example

- Forward process: given parameters $x_1, x_2 \sim \mathcal{N}(0,1)$, observation y arises according to
$$y = x_1 + x_2 = g(x_1, x_2)$$
- Inverse $(x_1, x_2) = g^{-1}(\hat{y})$ for given observation $\hat{y}$ is undefined
  - Classical regularization: minimum norm solution $x_1 = x_2 = \frac{\hat{y}}{2}$ (disregards ambiguity!)
- Bayesian solution:
  - Introduce latent variable $z = x_1 - x_2 \sim \mathcal{N}(0,2)$
  - Reparametrize $p(x_1, x_2 \mid \hat{y})$ as $(x_1, x_2) = g_{\text{aug}}^{-1}(\hat{y}, z) = \left(\frac{\hat{y}+z^{(t)}}{2}, \frac{\hat{y}-z^{(t)}}{2}\right)$
    - For $t \in 1, \dots, T$:
      - Sample $z^{(t)} \sim \mathcal{N}(0,2)$ and compute $x_1^{(t)} = \frac{\hat{y}+z^{(t)}}{2}$ and $x_2^{(t)} = \frac{\hat{y}-z^{(t)}}{2}$
    - Return $\left\{\left(x_1^{(t)}, x_2^{(t)}\right)\right\}_{t=1}^{T}$ as a sample from the Bayesian posterior $p(x_1, x_2 \mid \hat{y})$

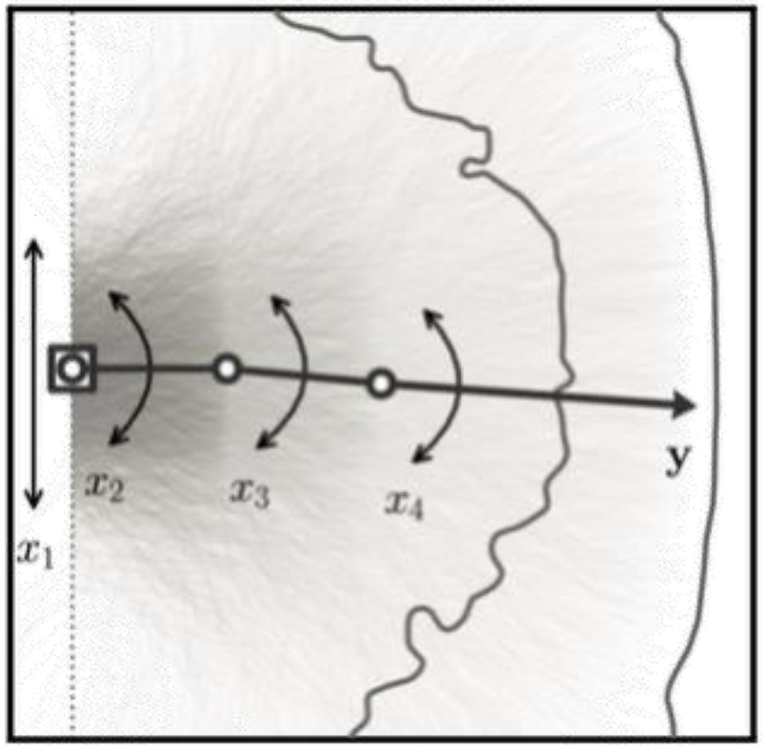
Generalize this to complex settings (non-linear s, noise, high dimensions) by INNs.

Non-linear Toy Example: Inverse Kinematics

Forward problem

- robot arm with 4 DOF: $x = [x_1, \dots, x_4]$
  - x_1: vertical position of first joint
  - x_2, x_3, x_4: joint angles
  - Gaussian priors: x_1 prefers the center
    x_2, x_3, x_4 prefer to be straight
- observation: hand position $y = [y_1, y_2]$
- geometric arm simulation $y = g(x)$
  implicitly defines likelihood $p(y \mid x)$

prior distribution $p(x)$



Non-linear Toy Example: Inverse Kinematics

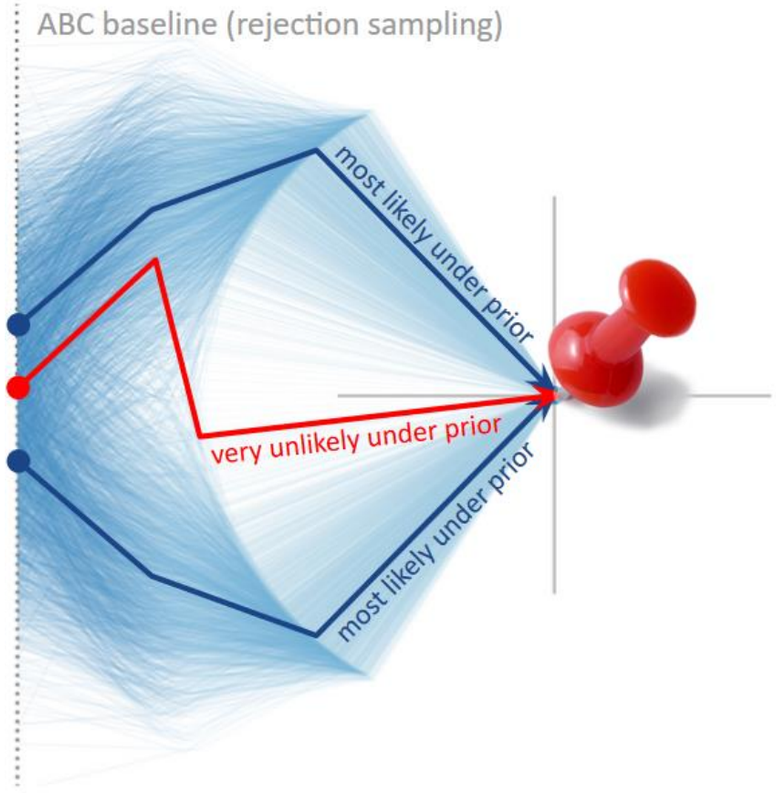
Forward problem

- robot arm with 4 DOF: $x = [x_1, \dots, x_4]$
  - x_1: vertical position of first joint
  - x_2, x_3, x_4: joint angles
  - Gaussian priors: x_1 prefers the center
    x_2, x_3, x_4 prefer to be straight
- observation: hand position $y = [y_1, y_2]$
- geometric arm simulation $y = g(x)$
  implicitly defines likelihood $p(y \mid x)$

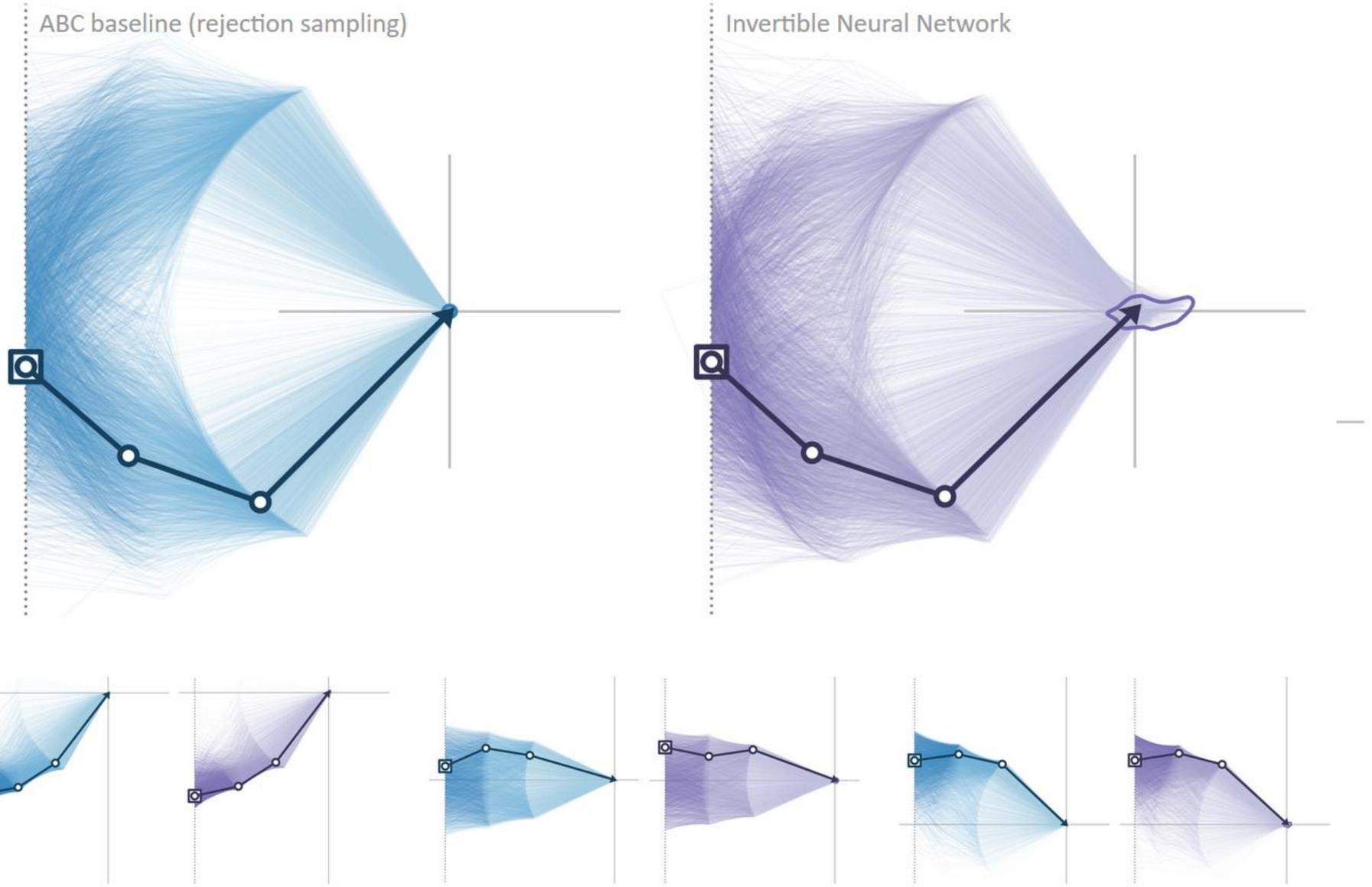
INN infers posterior $p(x \mid \hat{y})$ for given hand position $\hat{y}$

- For $t \in 1, \dots, T$:
  Sample $z^{(t)} \sim \mathcal{N}(0, \mathbb{I})$ and compute $x^{(t)} = f_\theta(\hat{y}, z^{(t)})$

posterior distribution $\mathrm{p}^*(x|\hat{y})$

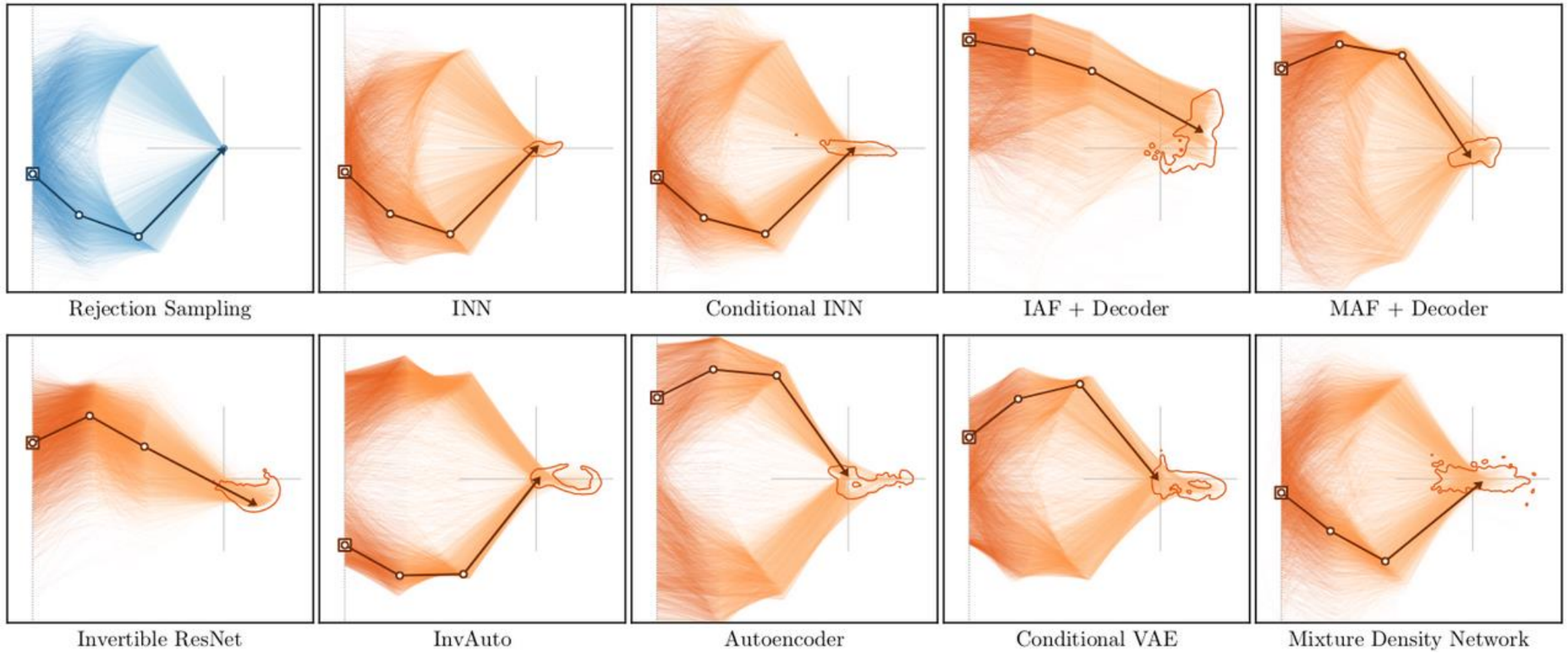


Non-linear Toy Example: Inverse Kinematics

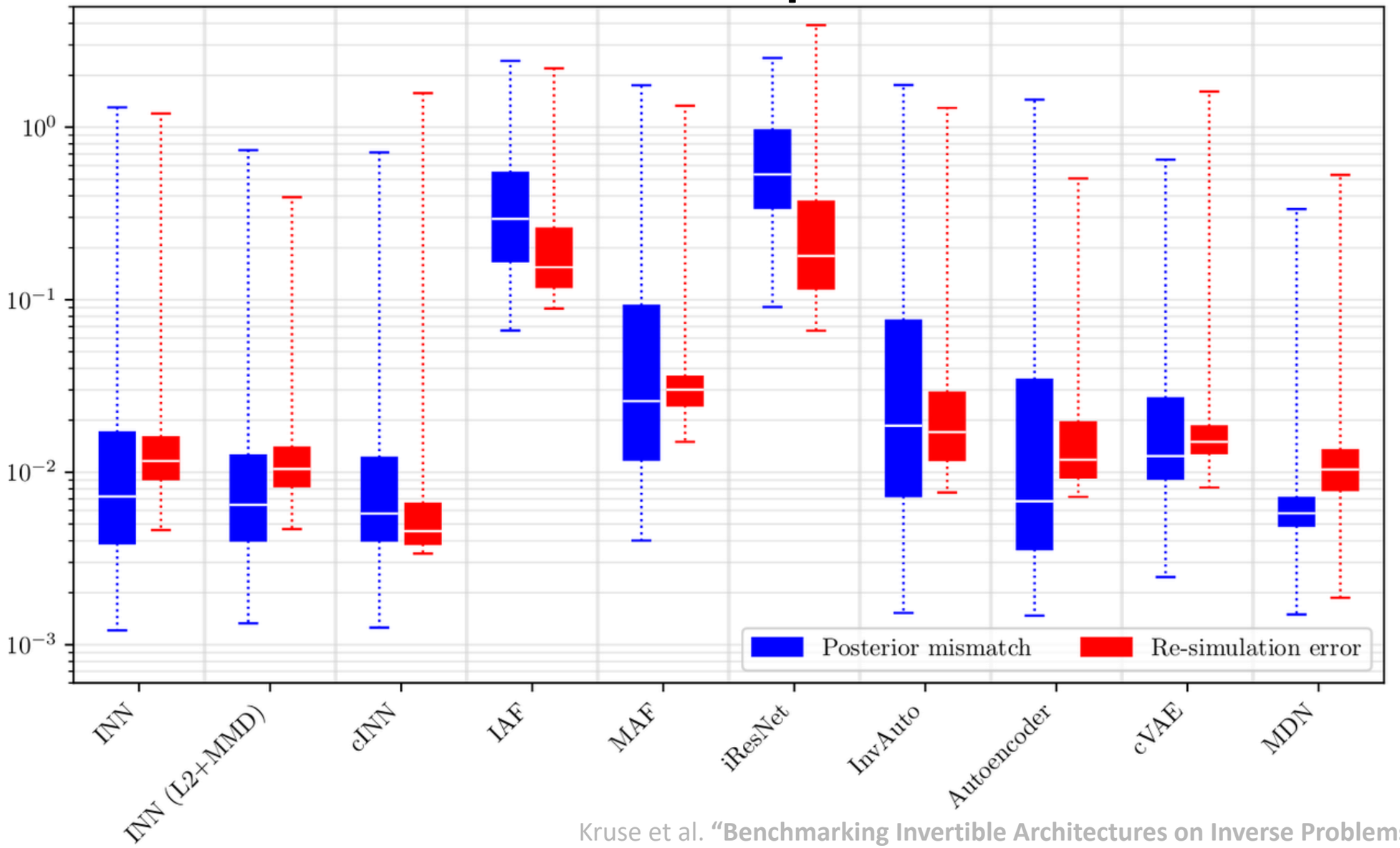


Comparison of Invertible Architectures
Kinematics Example

- Gound-truth posterior can be approximated with rejection sampling ("Approximate Bayesian Computation" – ABC)
- Performance metrics:
  - MMD between estimated and ground-truth posterior
  - re-simulation error $\|s(\hat{x}) - \hat{y}\|_2^2$: where does $s(\hat{x})$ actually land?



Comparison of Invertible Architectures
Kinematics Example



Kruse et al. **"Benchmarking Invertible Architectures on Inverse Problems"**, INNF 2019.

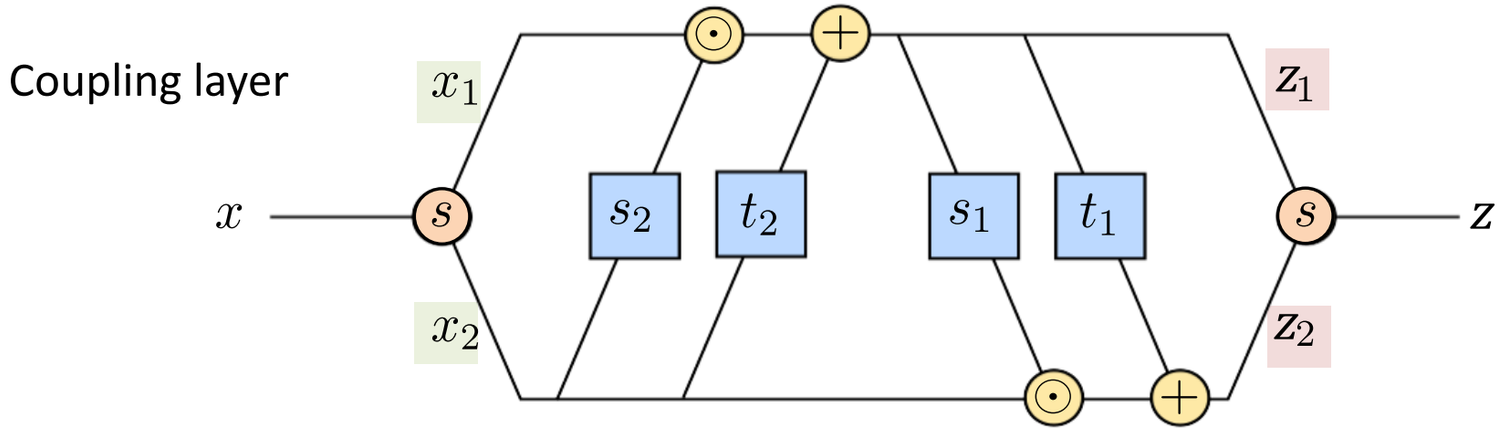
Invertible Neural Networks with Coupling Layers

We work mainly with RealNVP („non-volume preserving“) [Dinh et al. 2017]

- network is a sequence of *coupling layers*
- each coupling layer splits its input $x \in \mathbb{R}^D$ into two halves $x_1, x_2 \in \mathbb{R}^{D/2}$
- each half is subjected to an affine transformation ⇨ outputs $z_1, z_2 \in \mathbb{R}^{D/2}$
- affine coefficients are computed by standard fully connected or convolutional networks $s_{1,2} \in \mathbb{R}_+^{D/2}$ and $t_{1,2} \in \mathbb{R}^{D/2}$ from the other half's data

Forward computation: $z_1 = x_1 \odot s_2(x_2) + t_2(x_2), \quad z_2 = x_2 \odot s_1(z_1) + t_1(z_1)$

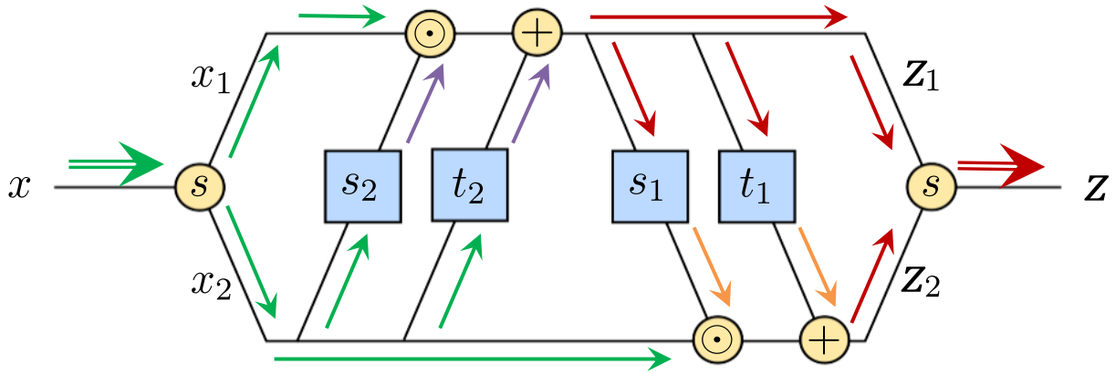
Inverse computation: $x_2 = z_2 \oslash s_1(z_1) - t_1(z_1), \quad x_1 = z_1 \oslash s_2(x_2) - t_2(x_2)$



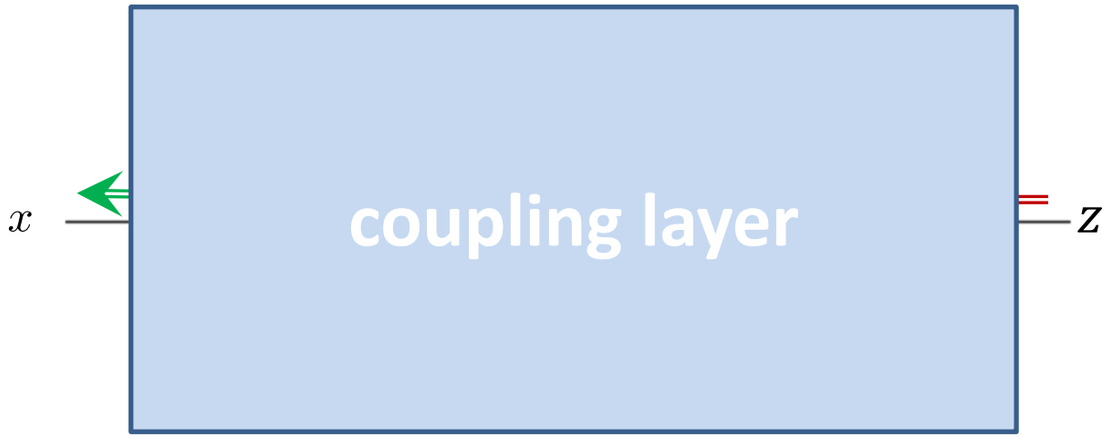
Invertible Neural Networks with Coupling Layers

- Data flow in a coupling layer

forward



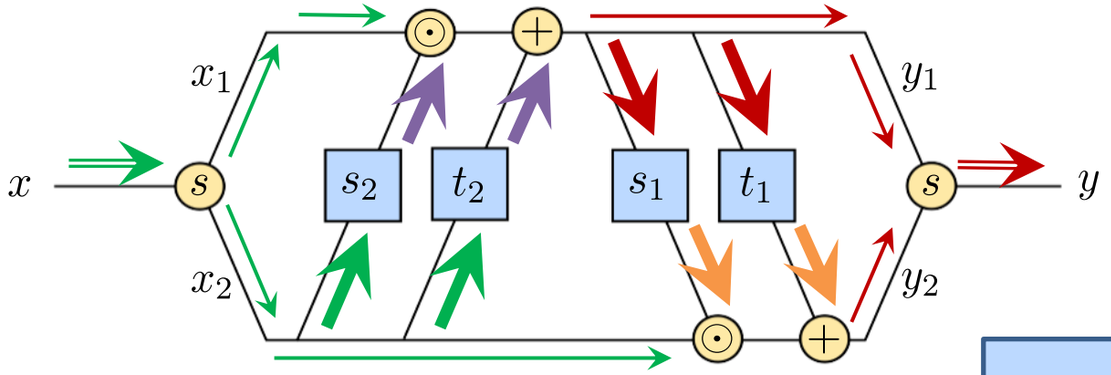
backward



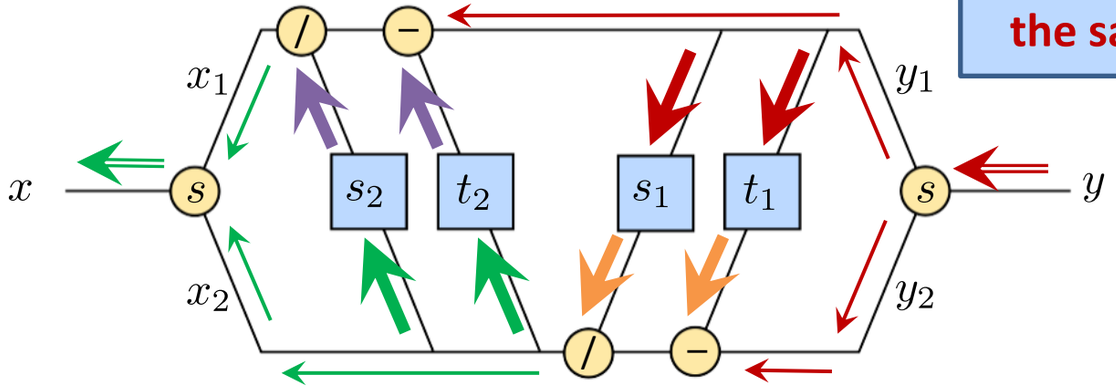
Invertible Neural Networks with Coupling Layers

- Data flow in a coupling layer

forward



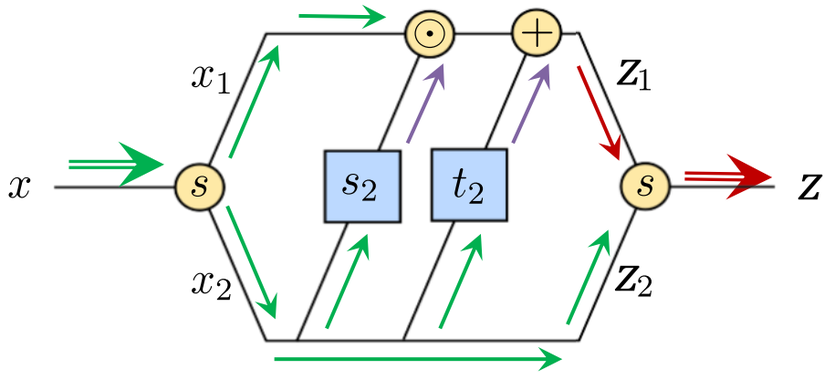
backward



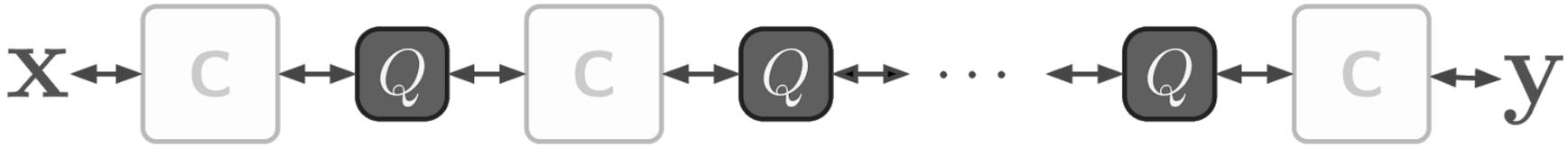
nested networks $s_{1/2}$ and $t_{1/2}$ are always executed in the same direction

Invertible Neural Networks with Coupling Layers

- Simplification: only one coupling per layer



- Recover the reverse coupling by stacking such layers into a deep INN:
  - alternate between coupling layers C
  - and permutation layers Q (random permutations are good enough in practice, learning Q is not necessary)

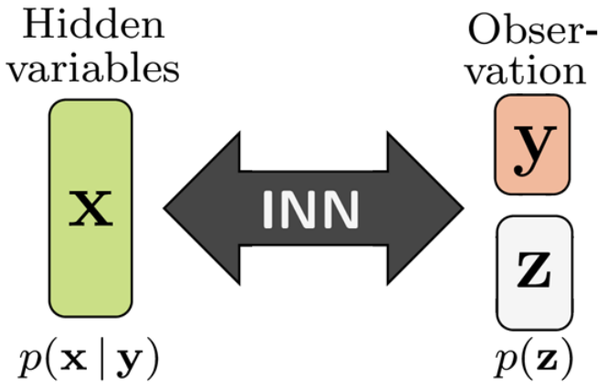


INN Architectures for Inverse Bayesian Inference

Augmented latent space

training: $(y, z) = f_\theta(x)$
s.t. $p(z) = \mathcal{N}(0, \mathbb{I})$

inference: sample $z \sim \mathcal{N}(0, \mathbb{I})$
compute $x = f_\theta^{-1}(\hat{y}, z)$
⇨ $x \sim p(x \mid \hat{y})$

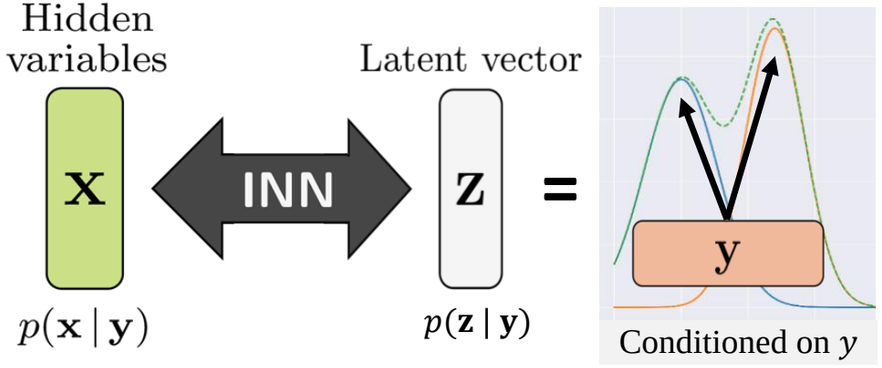


the original

Latent mixture INN

training: $z = f_\theta(x)$
s.t. $p(z) = \mathrm{GMM}(z; y) = \sum_y \mathcal{N}(\mu_y, \Sigma_y)$

inference: sample $z \sim \mathcal{N}(\mu_{\hat{y}}, \Sigma_{\hat{y}})$
compute $x = f_\theta^{-1}(z)$
⇨ $x \sim p(x \mid \hat{y})$

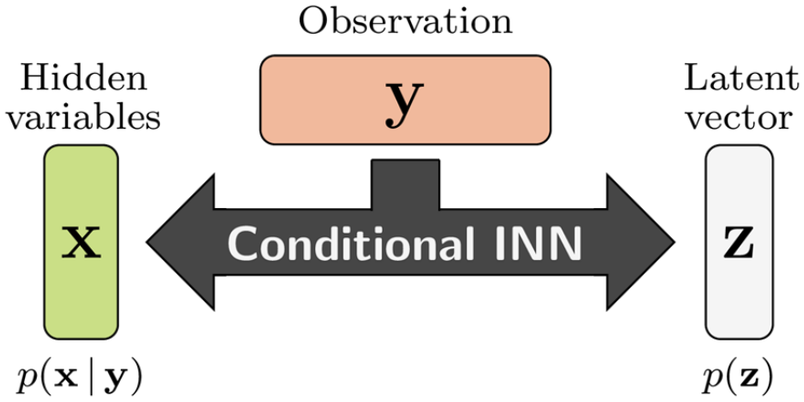


disentanglement

Conditional INN

training: $z = f_\theta(x; y)$
s.t. $p(z) = \mathcal{N}(0, \mathbb{I})$

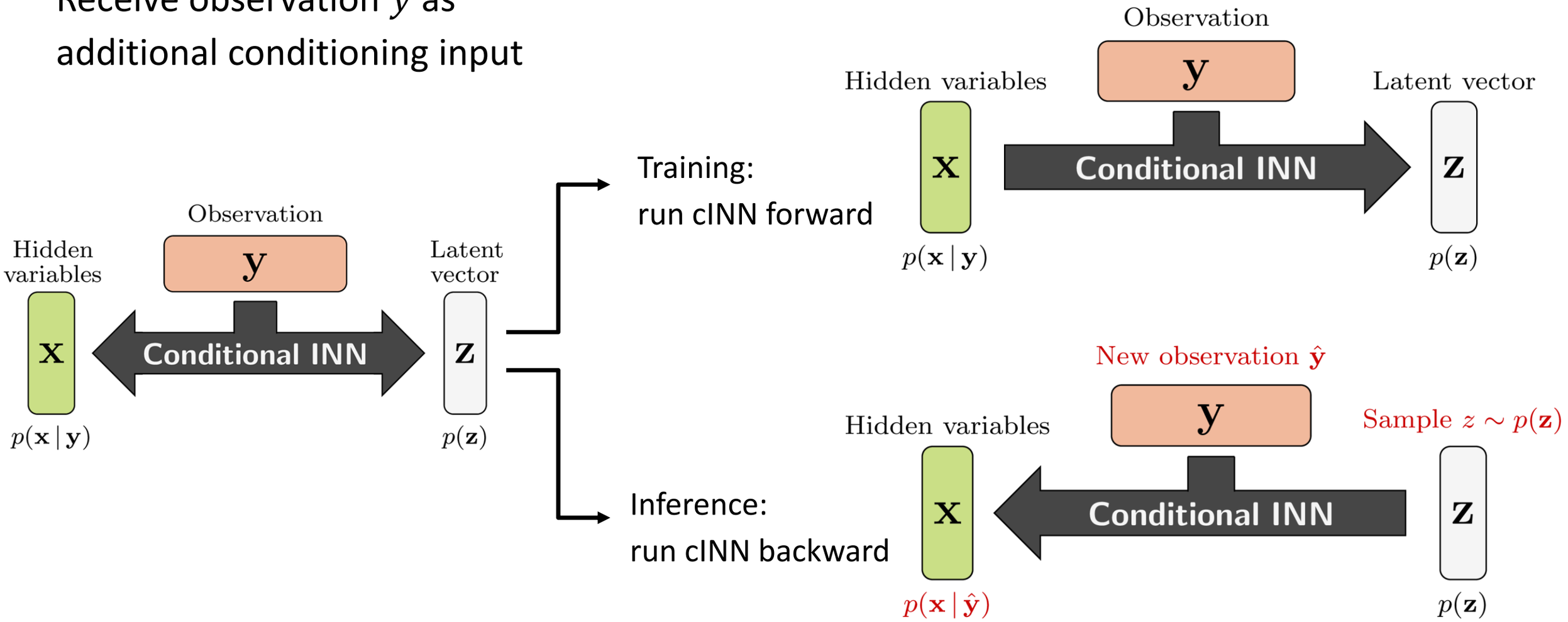
inference: sample $z \sim \mathcal{N}(0, \mathbb{I})$
compute $x = f_\theta^{-1}(z; \hat{y})$
⇨ $x \sim p(x \mid \hat{y})$



simulation-based inference

Conditional INN (cINN)

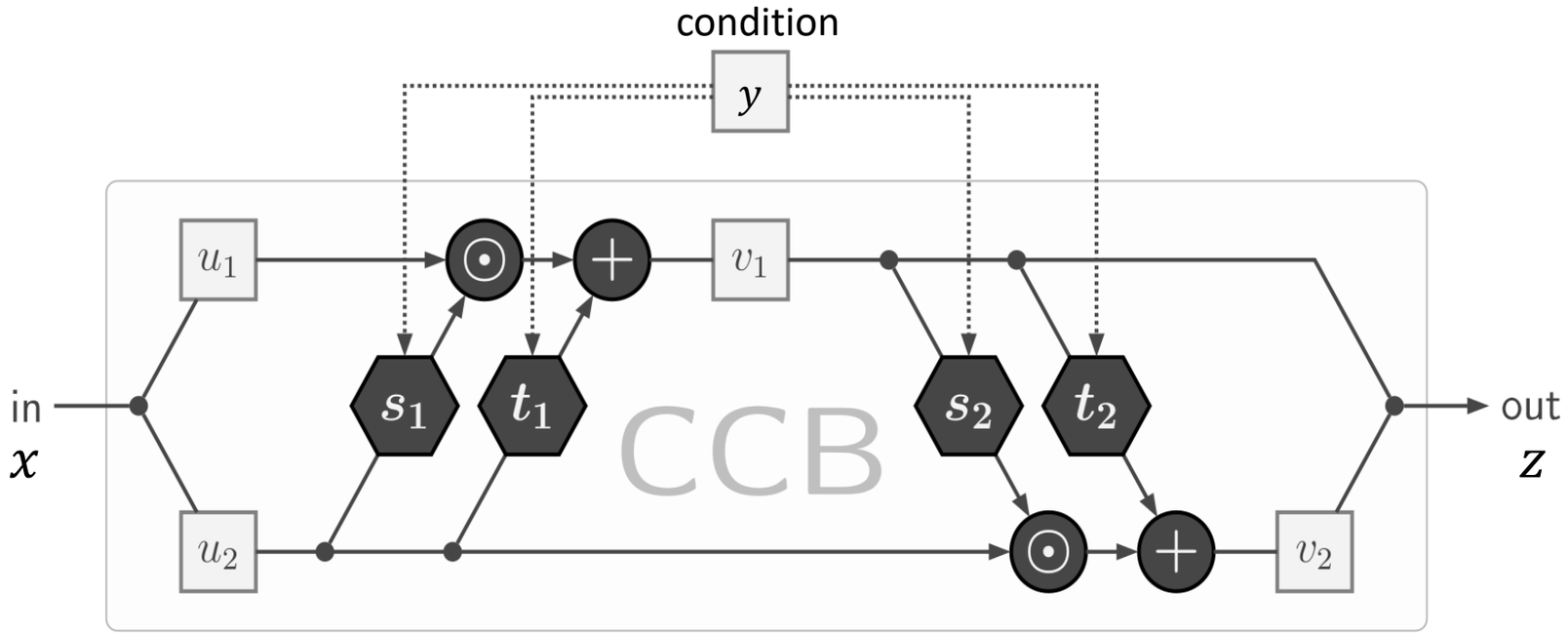
Receive observation y as additional conditioning input



cINN Architecture

Conditional coupling layers: simple generalization of standard coupling layers

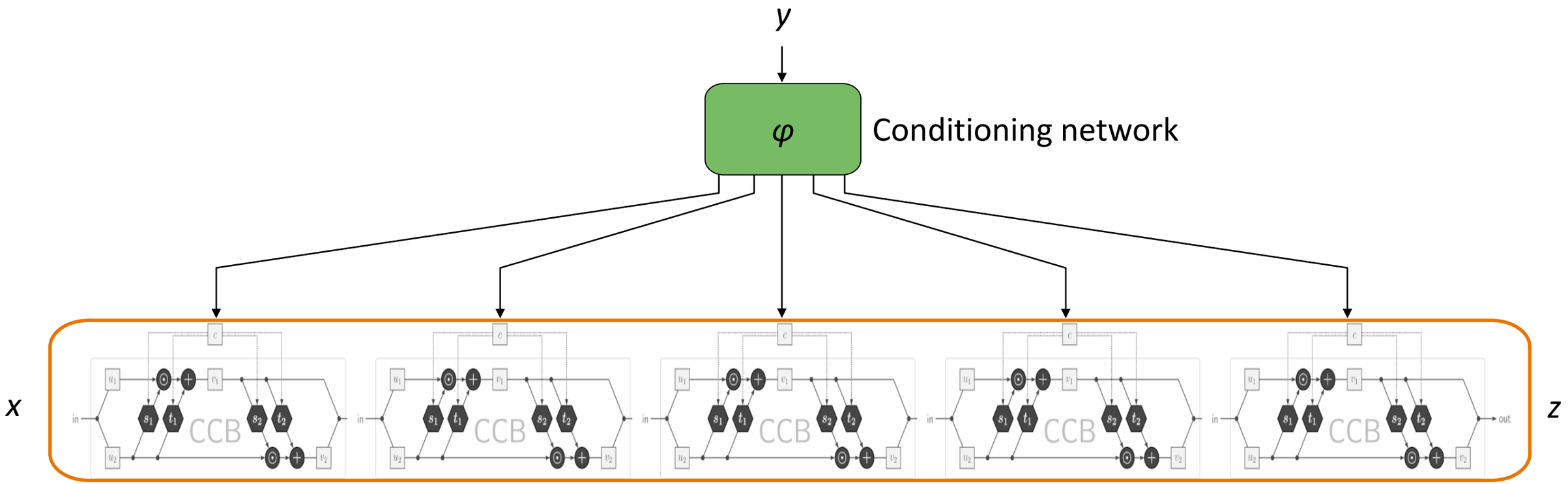
- Coefficient networks $s_{1,2}, t_{1,2}$ get additional conditioning input y
- For fixed condition, coupling blocks are still invertible (since coefficient networks are never inverted)
- Computes $\boldsymbol{z} = f(\mathbf{x}; \mathbf{y})$ and its inverse $\mathbf{x} = f^{-1}(\boldsymbol{z}; \mathbf{y})$



cINN Architecture

Conditional INN: sequence of conditional coupling layers

- If condition is complex (e.g. image): preprocessing via conditioning network $\phi(\mathbf{y})$ (= feature extraction)
- Computes $\boldsymbol{z} = f(\mathbf{x}; \boldsymbol{\phi}(\mathbf{y}))$ and its inverse $\mathbf{x} = f^{-1}(\boldsymbol{z}; \phi(\mathbf{y}))$



cINN Training

Estimated conditional density $\hat{p}(x \mid y)$ expressed via 'reparameterization trick'

- Let the cINN represent the function $z = f_\theta(x\,; y)$
- Train cINN such that $p_z(z) \approx \mathcal{N}(0, \mathbb{I})$
- Then $\hat{p}(x \mid y) \approx p^*(x \mid y)$ is defined by the change-of-variables formula

$$\hat{p}(x \mid y) = p_z\big(z = f_\theta(x\,; y)\big) \left| \det\left(\frac{\partial f_\theta(x\,; y)}{\partial x}\right)\right|$$

Can be trained with maximum likelihood loss:

$$\hat{\theta} = \arg\max_\theta \sum_{i \in \mathcal{TS}} \hat{p}(x_i \mid y_i)$$

$$= \arg\min_\theta \sum_{i \in \mathcal{TS}} \left(\frac{\|f_\theta(x_i; y_i)\|_2^2}{2} - \sum_{l=1}^{L} \log(\|s_l(x_i; y_i\|_1) \right)$$

Since f_θ is invertible (given $\hat{y}$), we get a generative model for free:

$$x \sim \hat{p}(x \mid \hat{y}) \quad \Leftrightarrow \quad z \sim \mathcal{N}(0, \mathbb{I}) \quad \text{and} \quad x = f_{\hat{\theta}}^{-1}(z; \hat{y})$$

BayesFlow: cINNs for Simulation-Based Inference

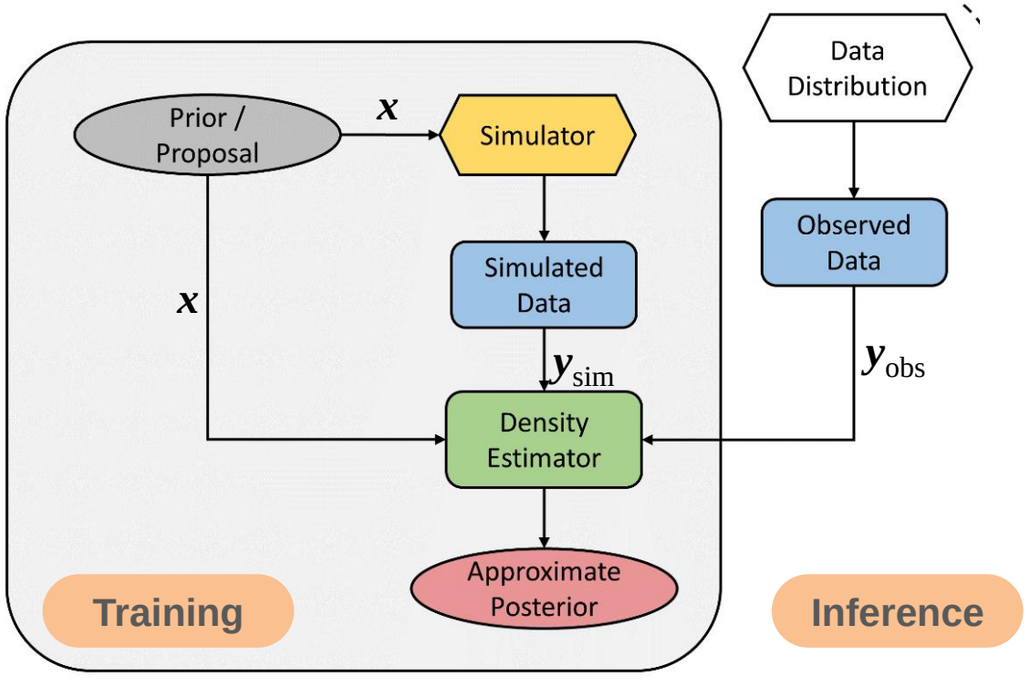
- Train an approximate posterior $\hat{p}(\boldsymbol{x} \mid \boldsymbol{y}) \approx p(\boldsymbol{x} \mid \boldsymbol{y})$ that works for any $\hat{\boldsymbol{y}} = \boldsymbol{y}_{\text{obs}}$
  - Expensive one-time (upfront) training of $\hat{p}$ using *simulated* data
  - Each inference query $\hat{p}(\boldsymbol{x} \mid \boldsymbol{y}_{\text{obs}})$ (for different $\boldsymbol{y}_{\text{obs}}$) is then cheap

  ⇨ Training effort quickly amortizes over multiple cheap queries

- Can be elegantly realized with cINNs: **BayesFlow**
  - Meta-algorithm:
    - Repeat: (training phase)
      - Simulate parameters $\boldsymbol{x} \sim p(\boldsymbol{x})$
      - Run the simulation $\boldsymbol{y}_{\text{sim}} = g(\boldsymbol{x};\ \boldsymbol{\xi})$
      - Perform maximum likelihood training of a cINN with (a batch of) $(\boldsymbol{x}, \boldsymbol{y}_{\text{sim}})$ pairs

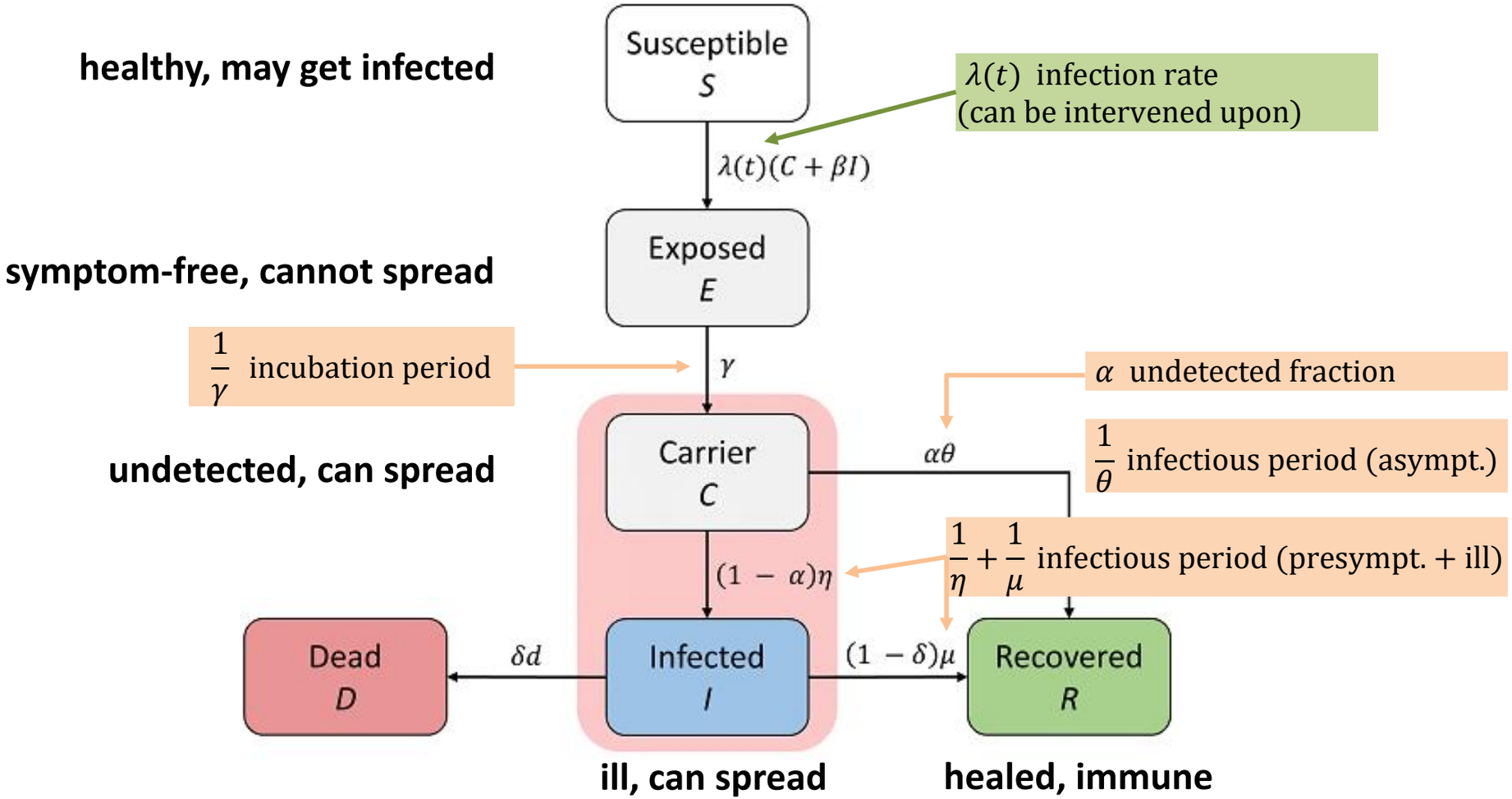
      Until convergence to $\hat{p}(\boldsymbol{x} \mid \boldsymbol{y}_{\text{sim}})$

    - For each observation $\boldsymbol{y}_{\text{obs}}$: (inference phase)
      - Use cINN to compute $\hat{p}(\boldsymbol{x} \mid \boldsymbol{y}_{\text{obs}})$ or create sample $\{\boldsymbol{x}_i \sim \hat{p}(\boldsymbol{x} \mid \boldsymbol{y}_{\text{obs}})\}$



BayesFlow for Epidemiology: Covid-19 Forward Model

- Forward model: SECIRD compartmental model (Lotka-Volterra type ODE system):



BayesFlow for Epidemiology: Covid-19 Forward Model

- Forward model: Lotka-Volterra type ODE system, e.g. SECIRD:

healthy, may get infected

$$\frac{dS}{dt} = -\lambda(t)\left(\frac{C+\beta I}{N}\right)S$$

symptom-free, cannot spread

$$\frac{dE}{dt} = \lambda(t)\left(\frac{C+\beta I}{N}\right)S - \gamma E$$

undetected, can spread

$$\frac{dC}{dt} = \gamma E - (1-\alpha)\eta C - \alpha\theta C$$

ill, can spread

$$\frac{dI}{dt} = (1-\alpha)\eta C - (1-\delta)\mu I - \delta d I$$

healed, immune

$$\frac{dR}{dt} = \alpha\theta C + (1-\delta)\mu I$$

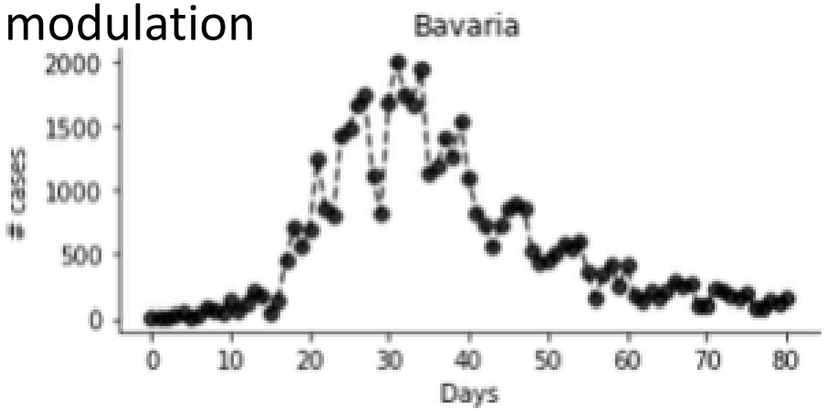
dead

$$\frac{dD}{dt} = \delta d I$$

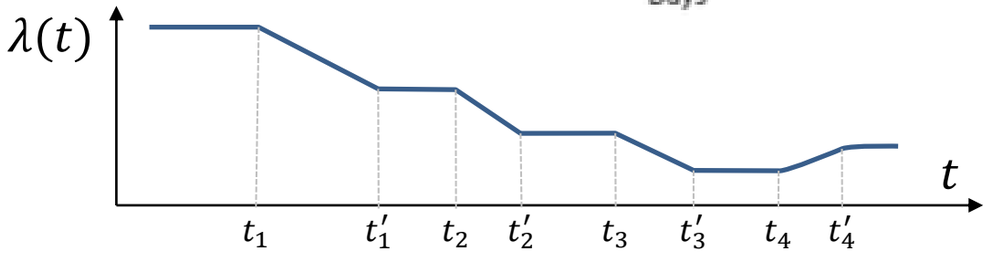
BayesFlow for Epidemiology: Enhanced Covid-19 Forward Model

- Enhance realism
  - Observation model: reporting delay, noise, weekly modulation

$$\frac{dI^{(obs)}}{dt} = (1 - f_I)\,(1 - \alpha)\,\eta\,C + \sqrt{I^{(obs)}}\sigma_I \frac{d\xi}{dt}$$
$$\frac{dR^{(obs)}}{dt} = (1 - f_R)\,(1 - \delta)\,\mu\,I + \sqrt{R^{(obs)}}\sigma_R \frac{d\xi}{dt}$$
$$\frac{dD^{(obs)}}{dt} = (1 - f_D)\,\delta\,d\,I + \sqrt{D^{(obs)}}\sigma_D \frac{d\xi}{dt}$$



  - Intervention model: four intervals where countermeasures were implemented or relaxed



  - Total: 34 parameters
    uninformative or very wide priors
  - Forward simulation is easy: sample from prior, solve ODEs

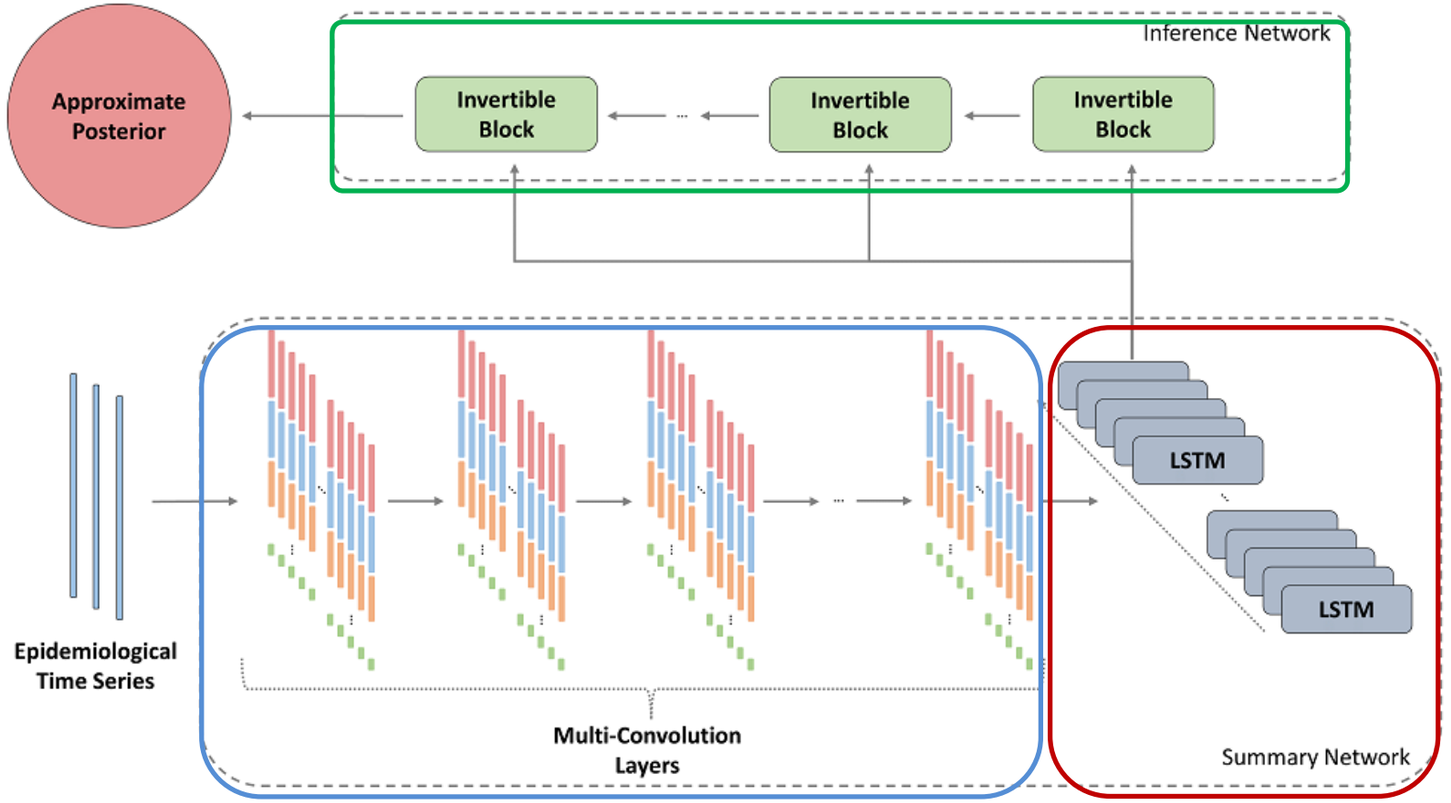
BayesFlow for Epidemiology: The Networks

- Inference problem: observation sequence (IRD) ⇨ parameter posteriors
  - Solve with BayesFlow network: cINN with statistical preprocessing networks for y

Convolutional:
- noise reduction
  feature detection

Recurrent (LSTM):
- variable-length sequence
  to fixed size summary

Invertible (cINN):
- posterior inference

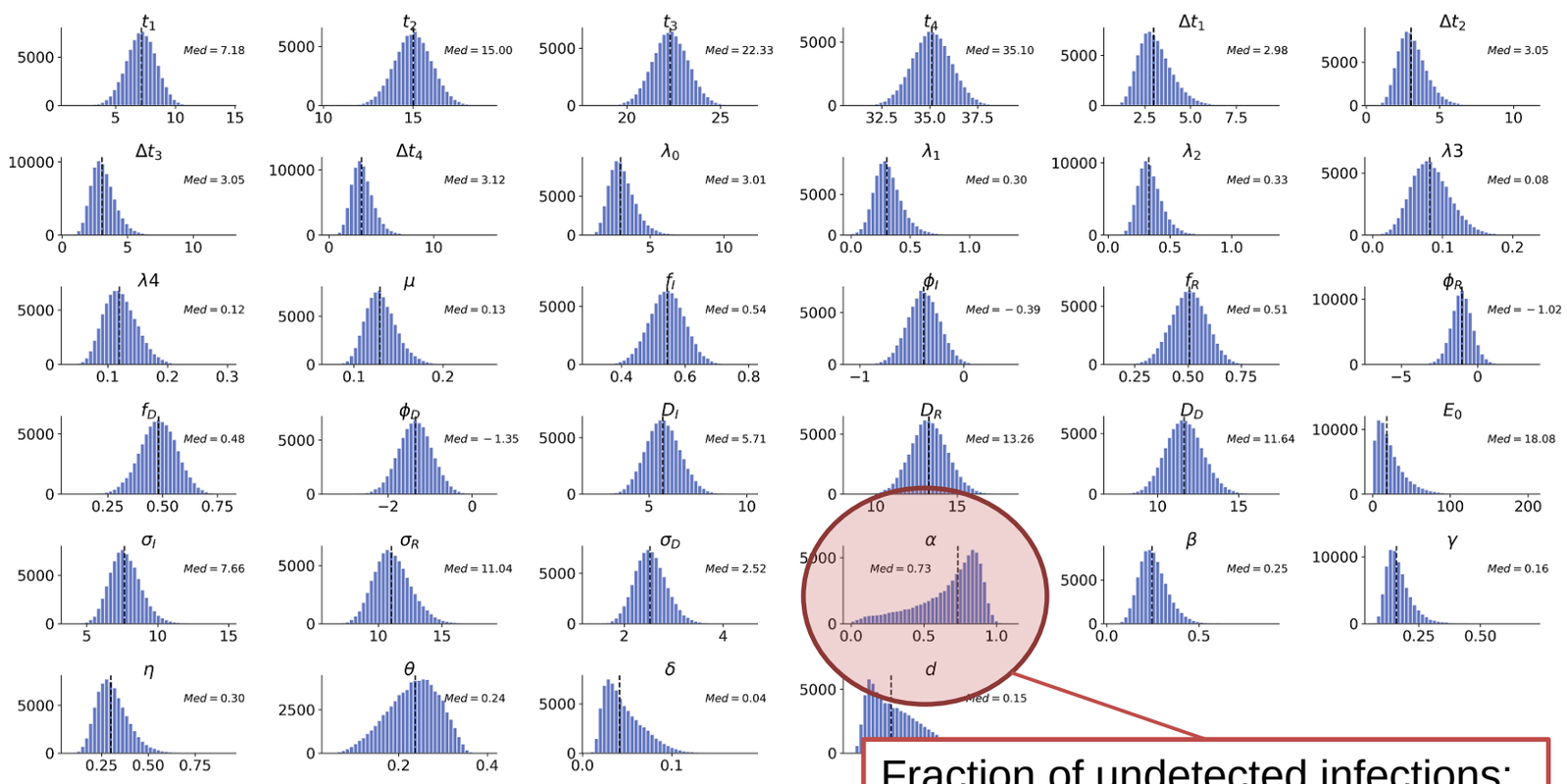


  - Training: end-to-end optimization of maximum likelihood loss with 50000-75000 simulations

BayesFlow for Epidemiology: Covid-19 Marginal Posteriors

Results: marginal posteriors for first wave in Germany (March – June 2020, 81 time steps)

- Fraction of infections remaining undetected:
  66% (median), 75% (mode)
- Serial interval: 9-10 days
- High likelihood to transmit disease *before* diagnosis
- time to recovery:
  4.6 days (undetected infections)
  11.3 days (diagnosed cases)
  (3.2 + 8.1 days before/after diagnosis)



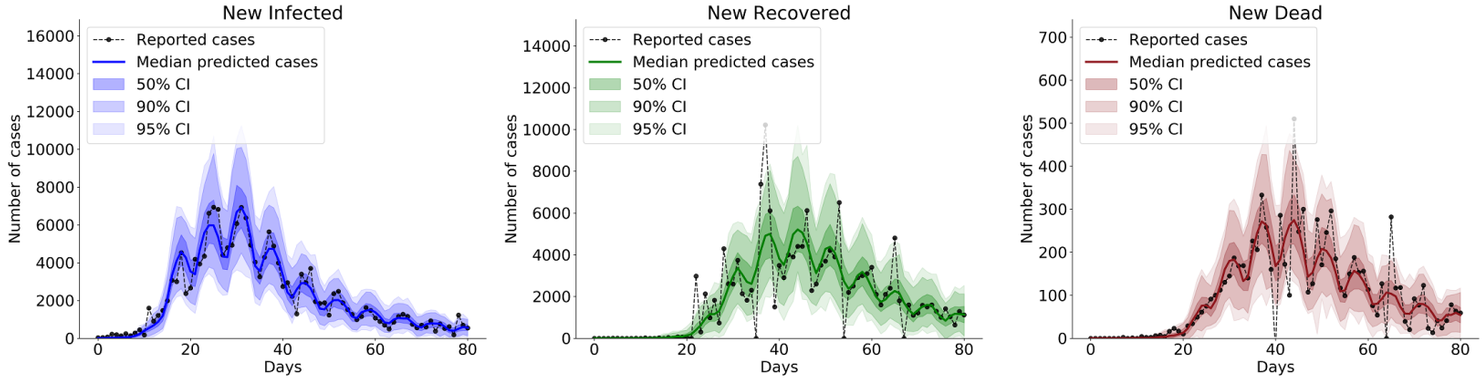
Fraction of undetected infections: uniform prior ⇨ peaked posterior

Correspond well to clinical findings

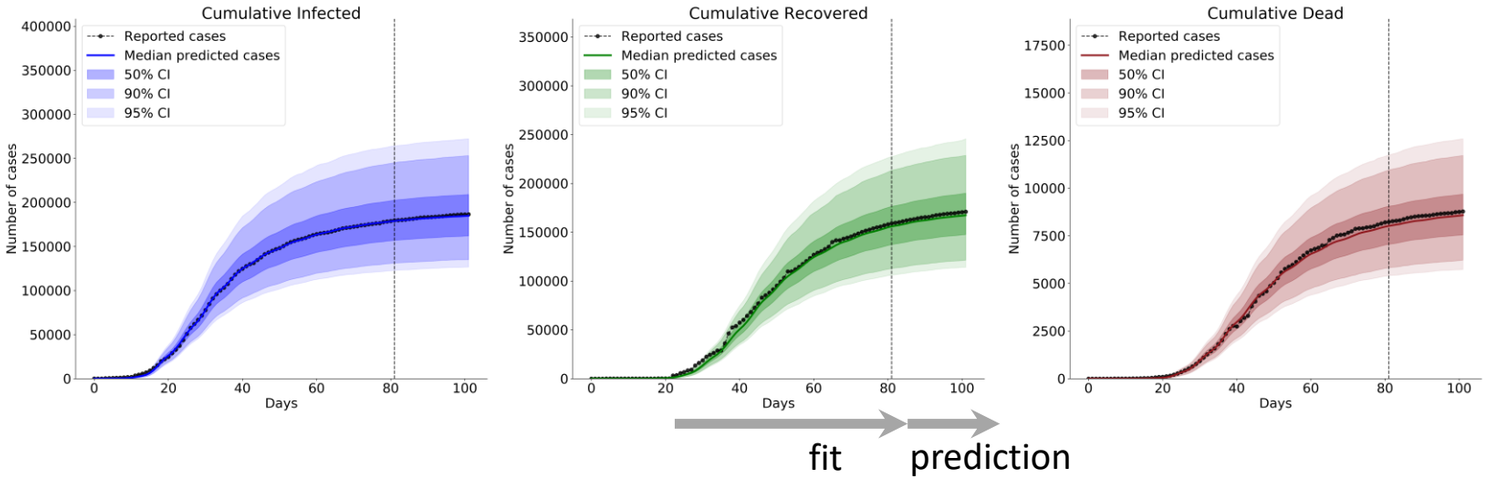
BayesFlow for Epidemiology: Covid-19 Predictive Posteriors

- Results: predictive posteriors for first wave in Germany (March – June 2020, 81 time steps)

daily:

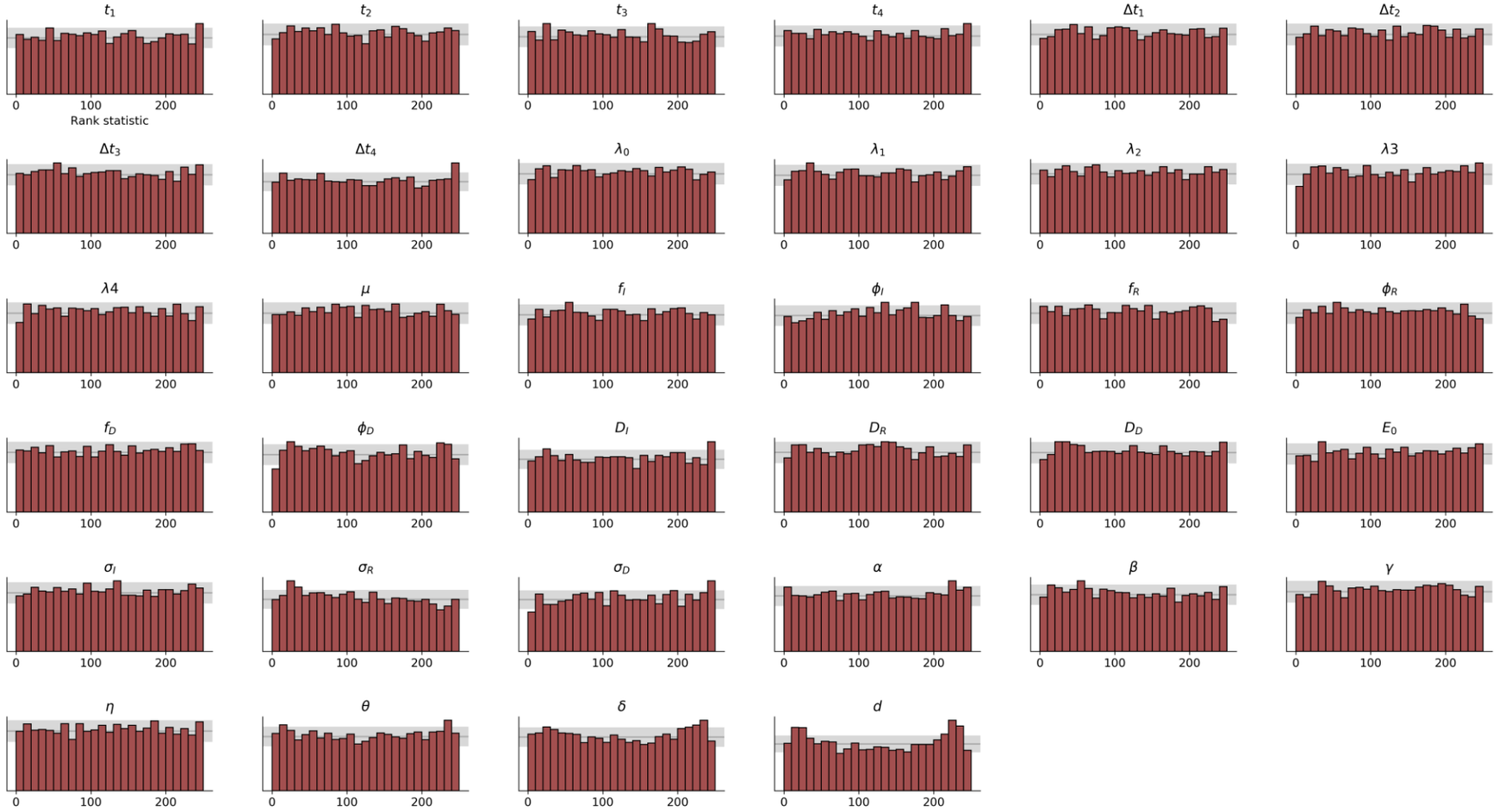


cumulated:



BayesFlow for Epidemiology: Covid-19 Uncertainty Calibration

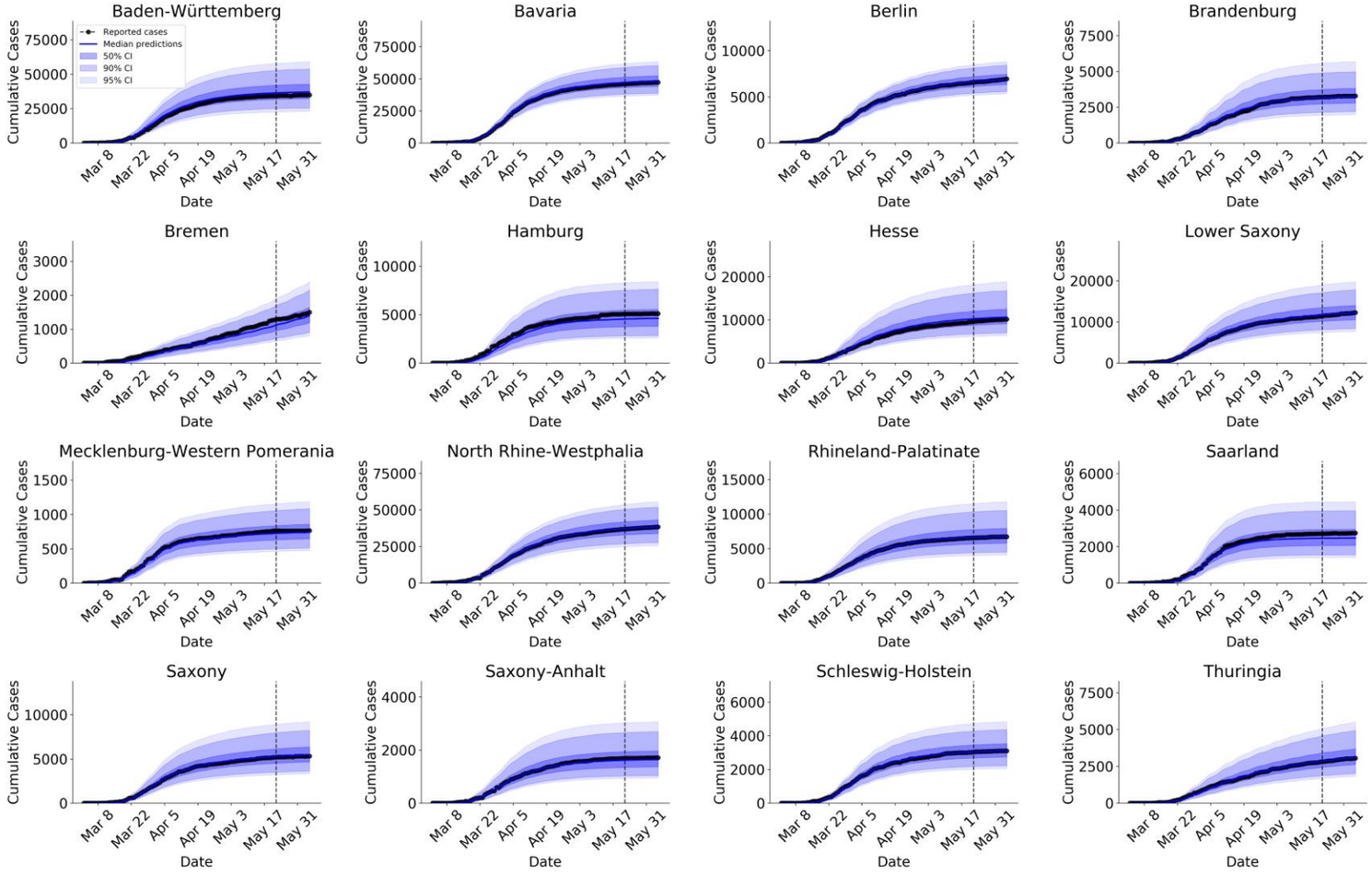
- Well-calibrated uncertainty quantification



Radev et al. **"Model-based Bayesian inference – an application to the COVID-19 pandemics in Germany"**, arXiv 2020.

BayesFlow for Epidemiology: Covid-19 Amortized Inference

- Amortized inference: same model works for German states (no re-training)

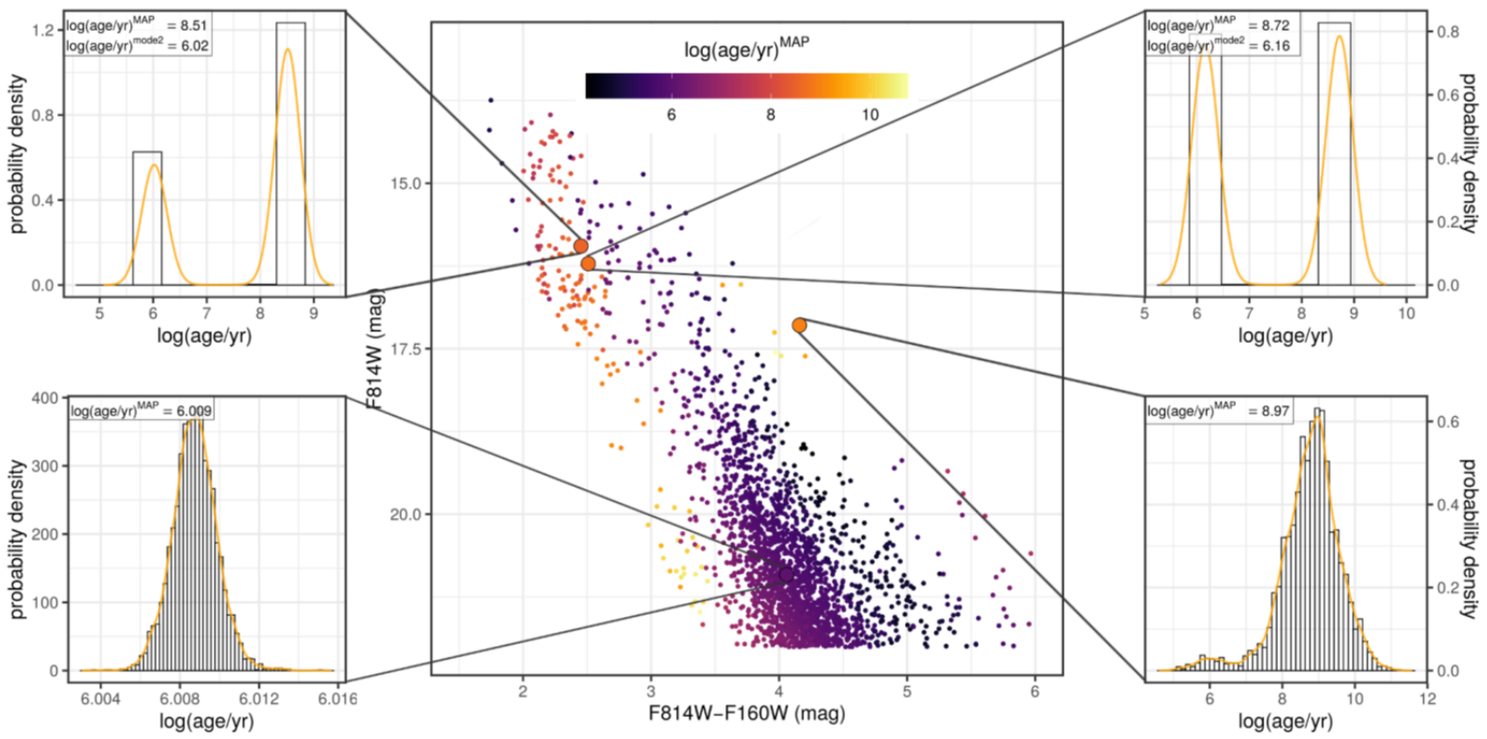


Radev et al. **"Model-based Bayesian inference – an application to the COVID-19 pandemics in Germany"**, arXiv 2020.

Application: Stellar Parameters Prediction from Photometry

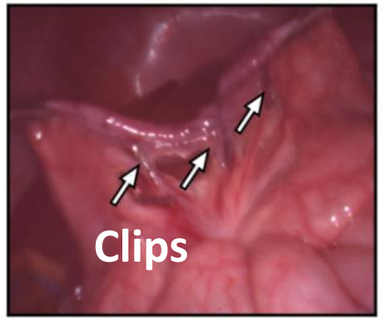
- Training with Stellar Evolution Model – PARSEC (Bressan et. al 2012)
  - Forward problem: simulate spectral properties from stellar parameters (mass, age, metallicity,…)
  - Inverse problem: infer stellar parameters from Hubble space telescope observations

- Results on Westerlund 2 young star forming cluster
  - Observations: color magnitude diagrams
  - Parameters: age, initial and current mass, luminosity, surface gravity, effective temperature
  - Well calibrated posteriors
  - MAP estimates close to truth
  - Physically plausible ambiguities: same spectral signal for young/heavy and older/lighter starts

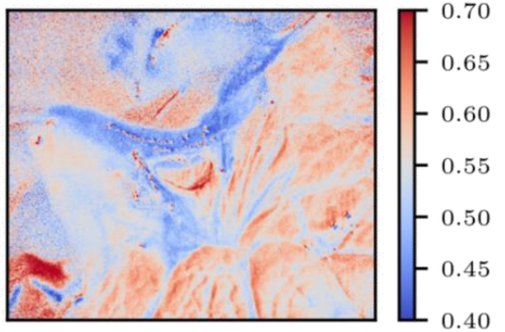


More Application Examples

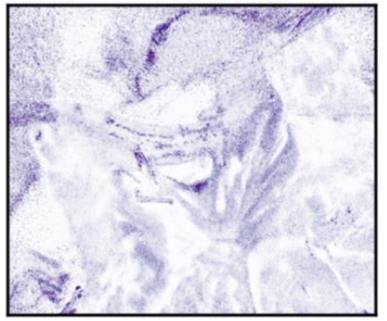
- Blood oxygenation detection with multi-spectral camera
  - Check correct clip placement before surgery
  - Check proper blood circulation at end of surgery
  - Select optimal camera



c) RGB image



a) Median s_{O_2}



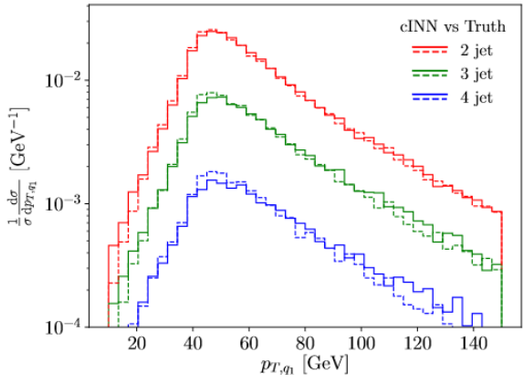
b) Est. uncertainty

- ZW-production unfolding at the LHC

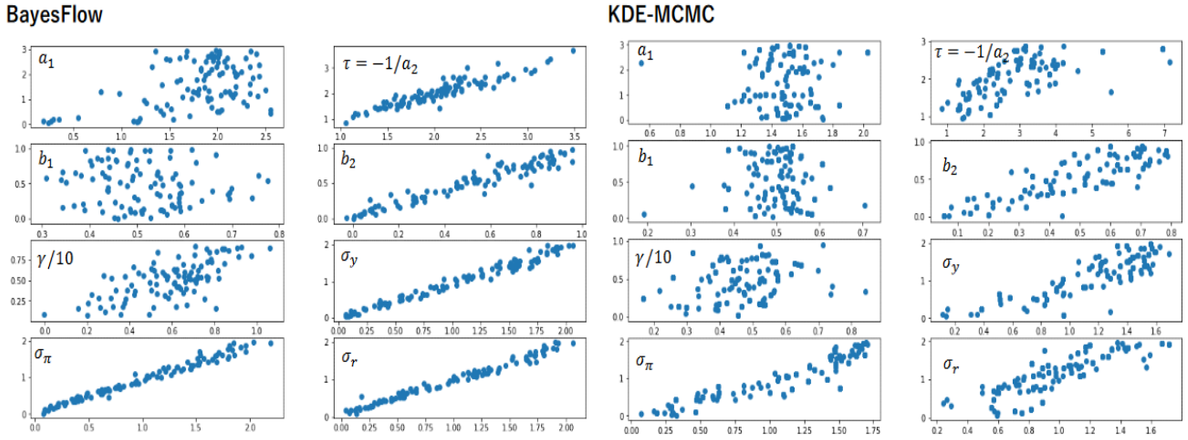
$$pp \to ZW^{\pm} + \text{jets} \to (\ell^{-}\ell^{+})\,(jj) + \text{jets}$$

  - can handle variable number of jets:

⇨ Talk tomorrow 14:40 by Anja Butter



- Agent-based models in finance / economy



Ardizzone et al. **"Analyzing inverse problems with invertible neural networks"**, ICLR 2019.
Bellagente et al. **"Invertible Networks or Partons to Detector and Back Again"**, arXiv, 2020.
Shiono "**Estimation of agent-based models using BayesFlow"**, SSRN, 2020.

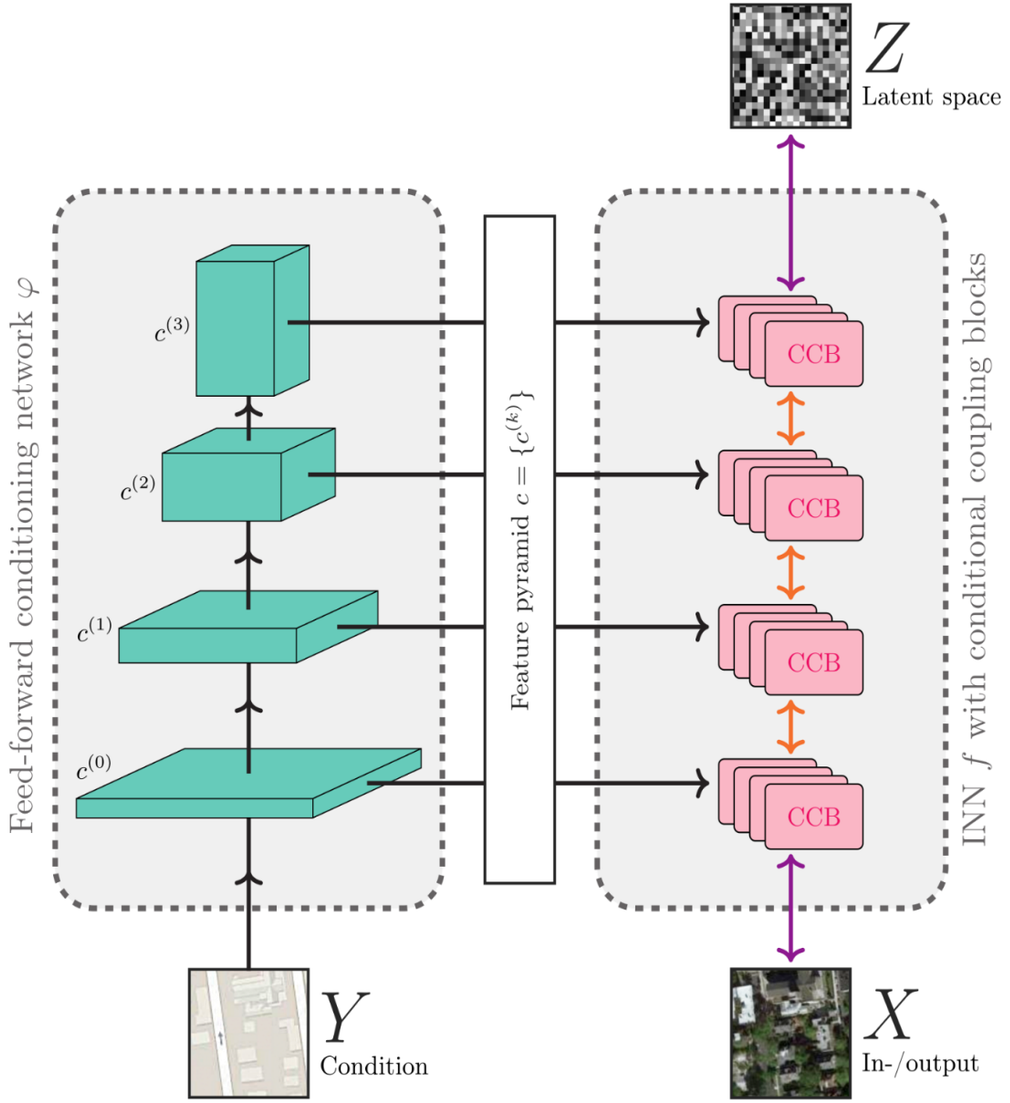
cINN for Image-to-Image Translation

- Example application: transform daylight images to night images
  - Condition y is the daylight image

  ⇨ preprocess with a feature-detection CNN
    - Layers of the CNN compute conditions $c^{(l)}$ at different resolutions / scales

  - Invertible network turns $p(z)$ into $p(x \mid y)$
    - Use multi-resolution features $c^{(l)}$ in the coupling block of corresponding resolution

  ⇨ creates diverse night images x for each y



Ardizzone et al. **"Guided Image Generation with Conditional Invertible Neural Networks"**, arXiv 2019.

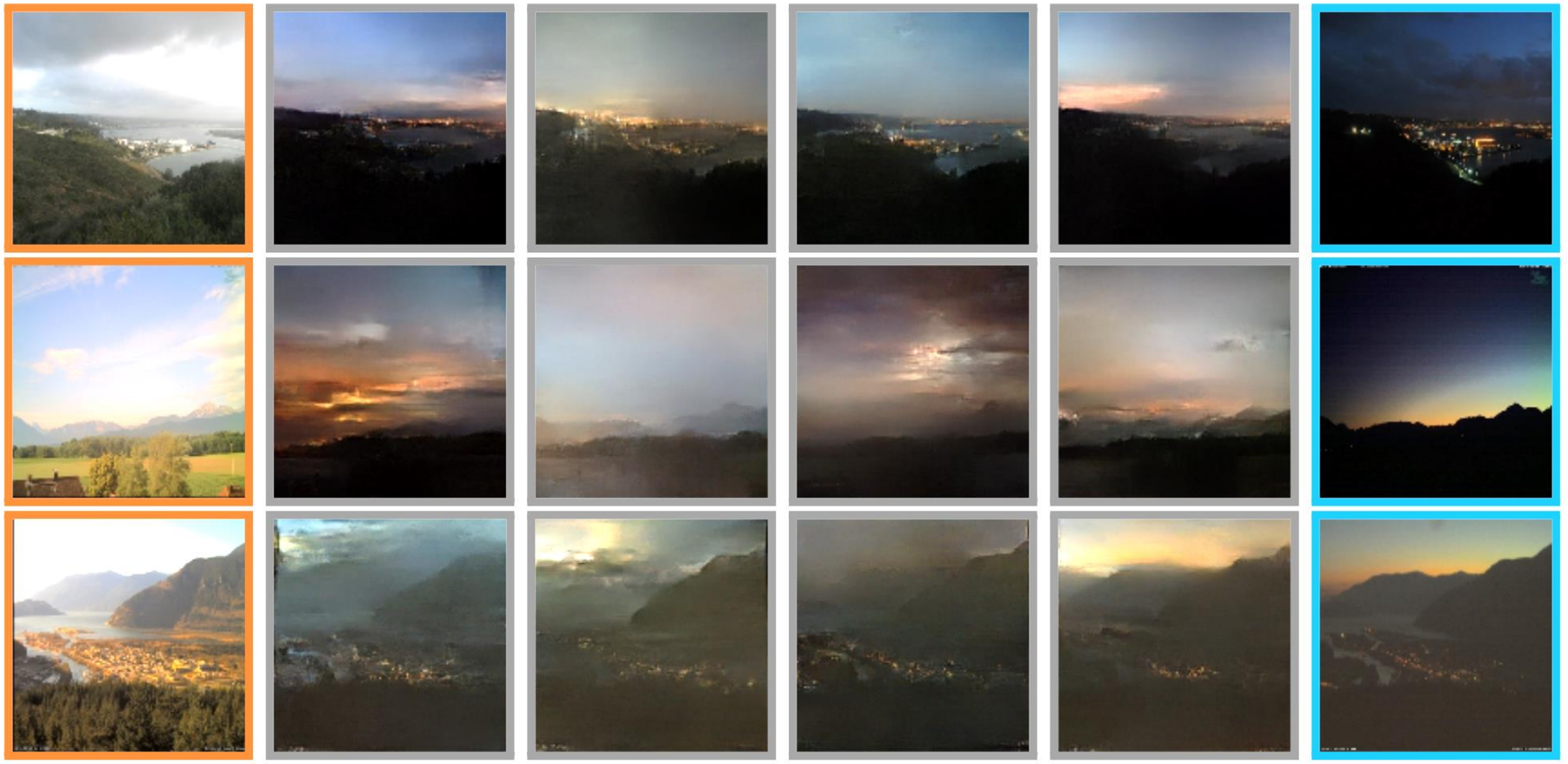
cINN for Image-to-Image Translation

- Results:

Condition *y*
Day image

Generated *x*
Night images

Ground truth *x*
Night image

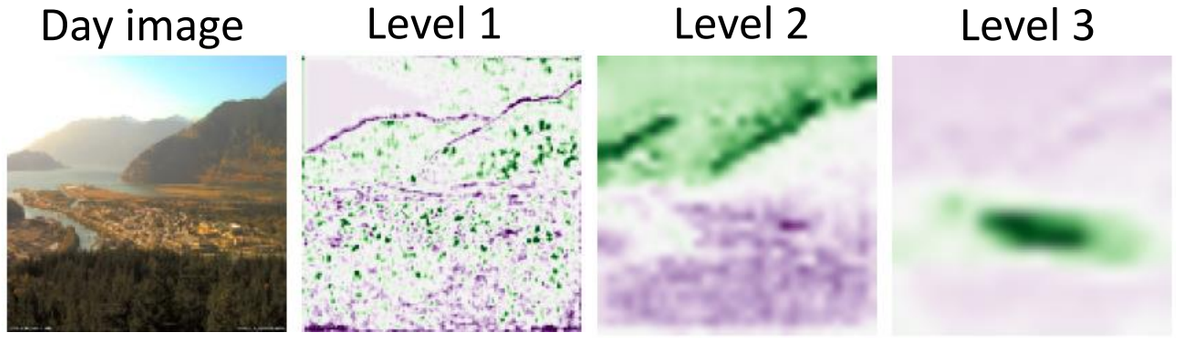


Ardizzone et al. **"Guided Image Generation with Conditional Invertible Neural Networks"**, arXiv 2019.

cINN for Image-to-Image Translation

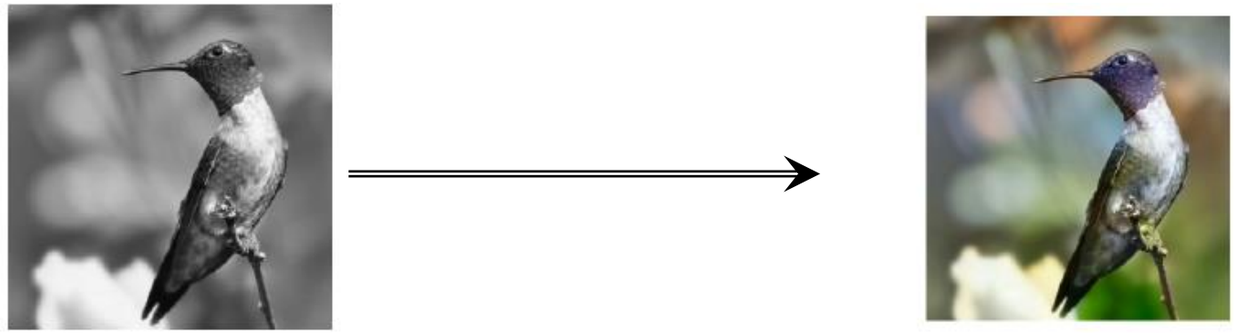
- Results:

- Multi-scale features learned by the conditioning network:
  - Level 1: edges and texture
  - Level 2: foreground / background
  - Level 3: populated areas (lights!)



Application: Colorization

- Inverse problem:
  - given grayscale image y (256×256, L-channel in Lab color space)
  - create **realistic** color channels $\hat{x} = [a, b]$ $(64 \times 64 \times 2)$



y

Application: Colorization

- Inverse problem:
  - given grayscale image y (256×256, L-channel in Lab color space)
  - create **realistic** color channels $\hat{x} = [a, b]$ $(64 \times 64 \times 2)$
  - which are **diverse** as z is varied

y

$z \sim p_z(z)$

$x \sim \hat{p}(x \mid y)$

  - Quiz: Which color image is the ground-truth?

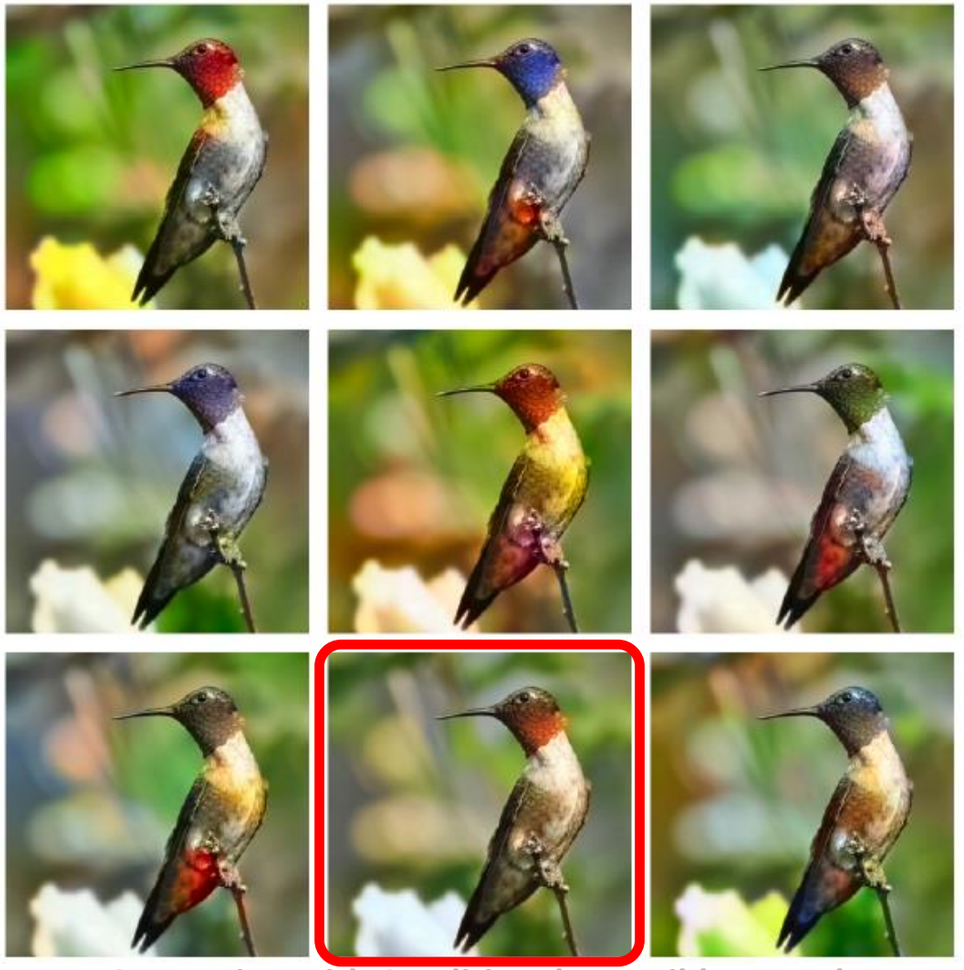
Application: Colorization

- Inverse problem:
  - given grayscale image y (256×256, L-channel in Lab color space)
  - create **realistic** color channels $\hat{x} = [a, b]$ $(64 \times 64 \times 2)$
  - which are **diverse** as z is varied

y

$z \sim p_z(z)$

$x \sim \hat{p}(x \mid y)$

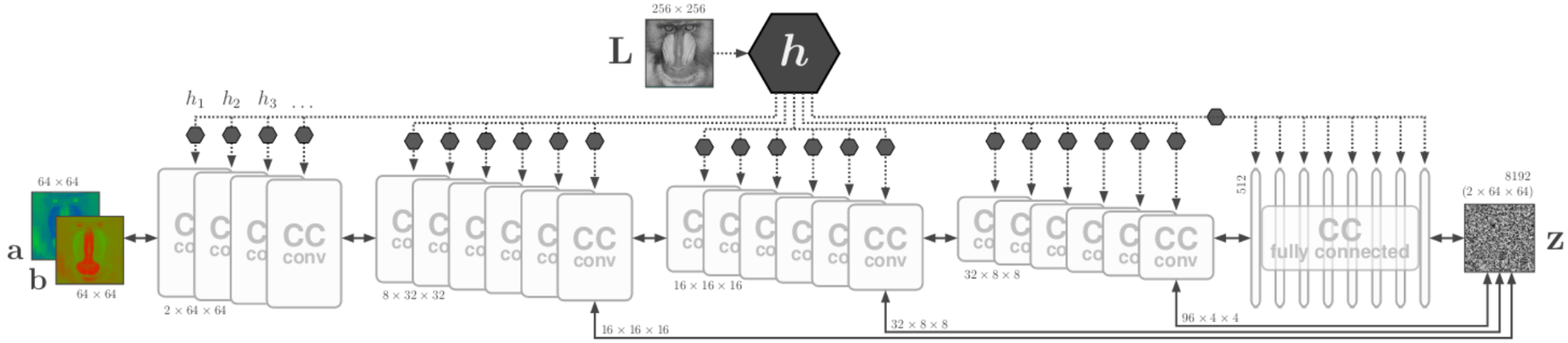


  - Quiz: Which color image is the ground-truth?

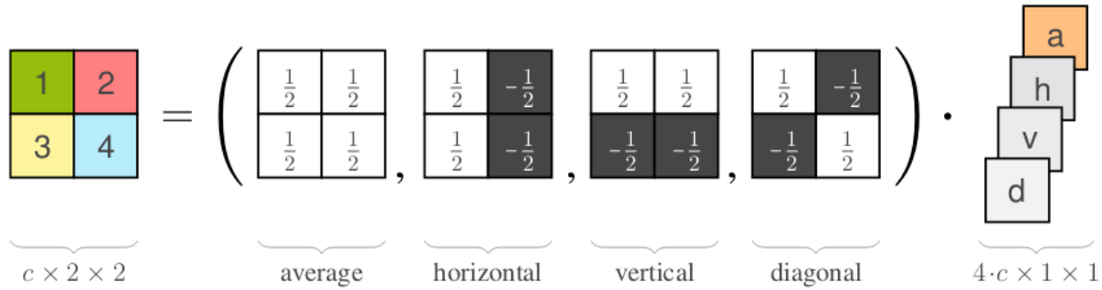
Ardizzone et al. **"Guided Image Generation with Conditional Invertible Neural Networks"**, arXiv 2019.

cINN Architecture for Colorization

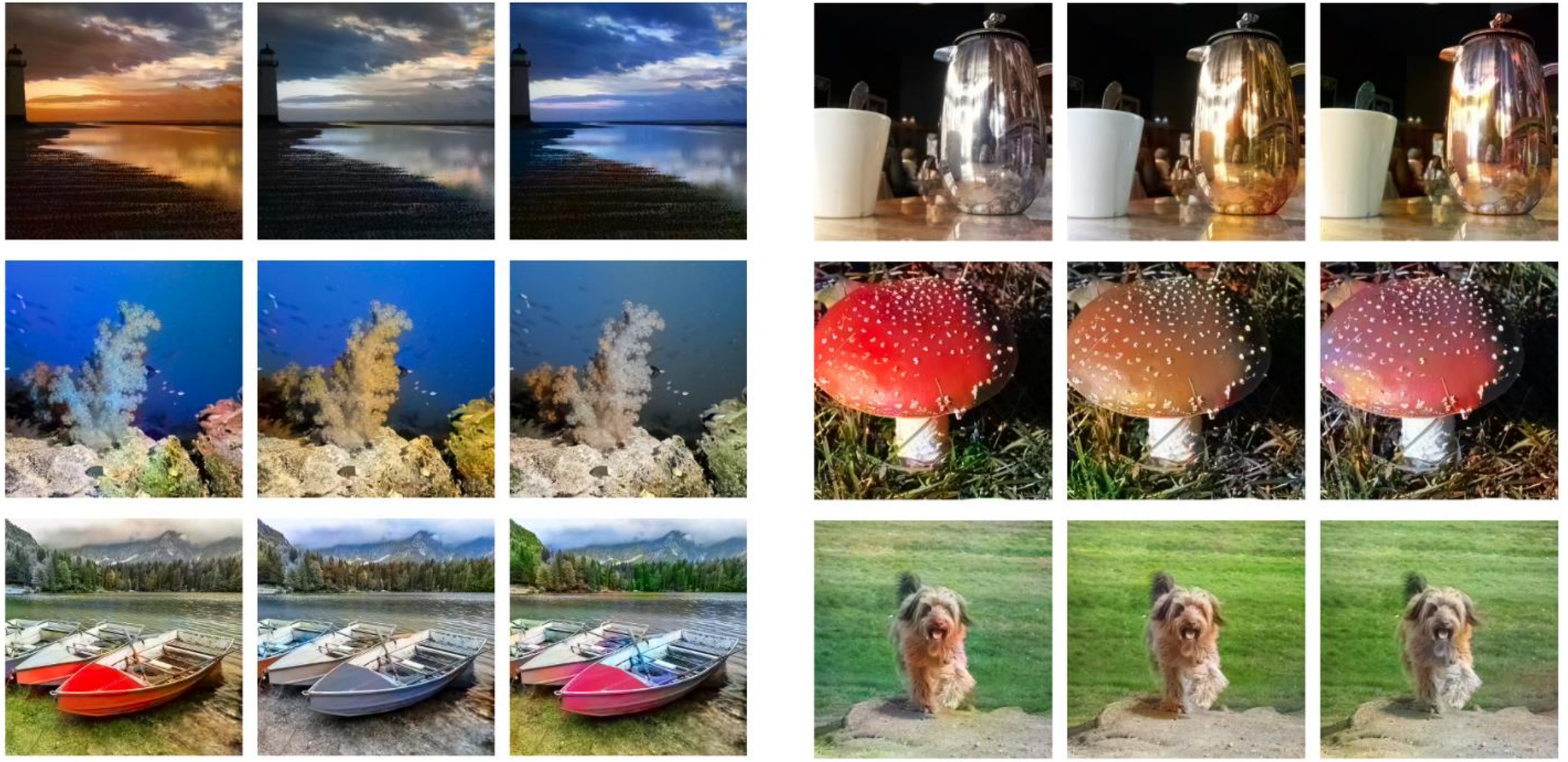
- Four convolutional stacks (with four to six coupling layers)
- Fully connected stack as backend (eight coupling layers)
- Coupling layers separated by random orthogonal matrices to mix channels
- Large feature detection network h (VGG), small conditioning networks h_l



- Multi-scale decomposition via *Haar-Wavelet* down-sampling (standard max pooling not invertible)



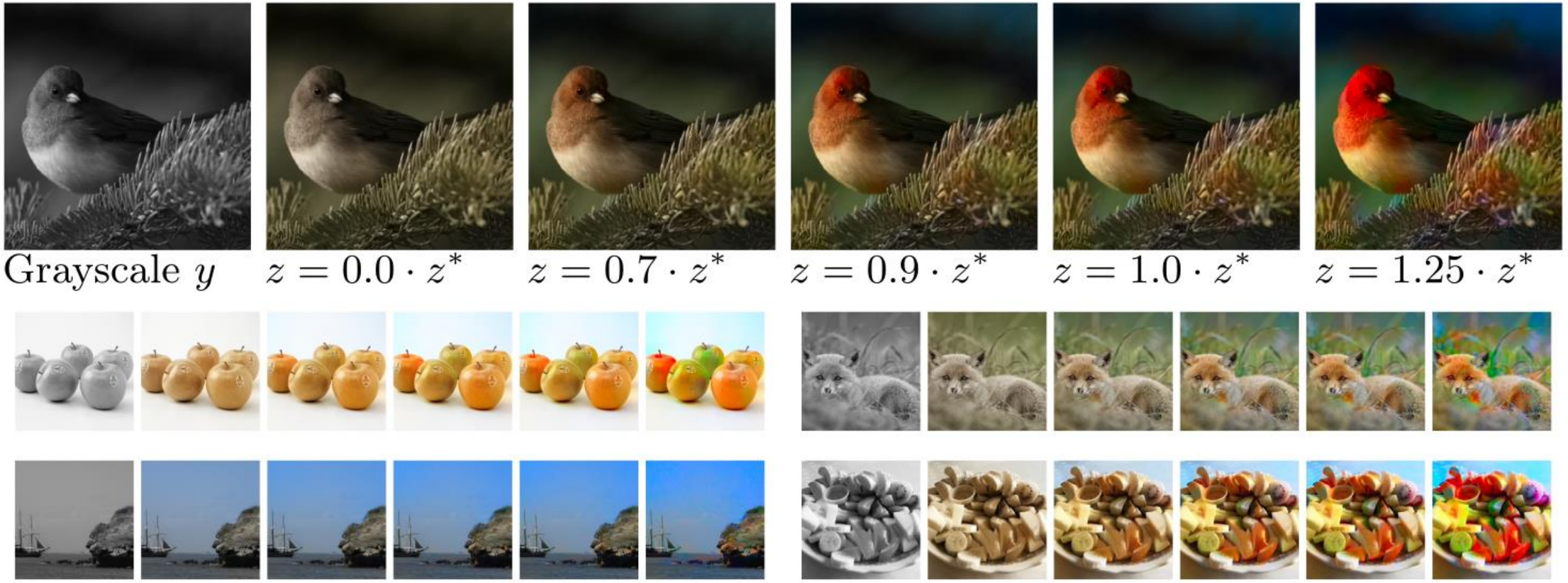
Colorization Examples



Ardizzone et al. **"Guided Image Generation with Conditional Invertible Neural Networks"**, arXiv 2019.

Colorization: Meaningful Latent Manipulations

- Magnitude of latent vector encodes color intensity
  - Linear interpolation from $z = 0$ outwards gradually increases saturation



Colorization: Meaningful Latent Manipulations

- Color transfer (in analogy to style transfer on MNIST)
  - Combining y from one image with true z from another transfers colors
    - Encode color image to latent space
    - Keep color encoding, but exchange condition (=grayscale image), then decode

Summary

Public code of our FrEIA library: https://github.com/VLL-HD/FrEIA

- INNs are very good density estimators:
  - Not yet quite as good as GANs (as trained by the Big Guys with 300 GPUs in parallel ☺)
  - But with much stronger mathematical interpretation and guarantees
  - Successful applications in particle and astro-physics, epidemiology, medicine, psychology
- Three main approaches to incorporate additional information
  - Conditional INN: learn $p_z(\mathbf{z} = f_{\text{INN}}(\mathbf{x}; \mathbf{y}))$
  - Latent mixture INN: learn $p_z(\mathbf{z} = f_{\text{INN}}(\mathbf{x}) \mid \mathbf{y})$
  - Augmented latent space INN: learn $p_{y,z}(\mathbf{y}, \mathbf{z} = f_{\text{INN}}(\mathbf{x}))$
  - We get the full posterior $p(x \mid y)$, both exactly and through samples
- Future work:
  - Improve architectures and training
  - Strengthen validation and mathematical guarantees
  - More applications in natural, social, and life sciences
  - Better incorporation of prior knowledge from the application domain

Thanks to our team and collaborators!

Visual Learning Lab, Uni Heidelberg:

Lynton Ardizzone
Jakob Kruse
Jens Müller
Felix Draxler
Radek Mackowiak
Peter Sorrenson
Carsten Rother

York University, Canada:

Marcus Brubaker

German Cancer Research Center, Heidelberg:

Tim Adler, Sebastian Wirkert, Lena Maier-Hein

Depart. of Physics and Astronomy, Uni Heidelberg:

Victor Ksoll, Ralf Klessen
Anja Butter, Armand Rousselot, Tilman Plehn

Inst. for Environmental Physics, Uni Heidelberg:

André Butz, Florian Kleinicke

Psychologisches Institut, Uni Heidelberg:

Stefan Radev, Ulf Mertens, Andreas Voss

References

Adler, T., et al.: “Uncertainty-aware performance assessment of optical imaging modalities with invertible neural networks”, Intl. J. Computer Assisted Radiology and Surgery 14(6):997–1007 (2019).

Ardizzone, L., et al.: “Analyzing inverse problems with invertible neural networks”, ICLR 2019, arXiv:1808.04730 (2018).

Ardizzone, L., Lüth, C., Kruse, J., Rother, C., & Köthe, U.: “Guided Image Generation with Conditional Invertible Neural Networks”, arXiv:1907.02392 (2019).

Ardizzone, L., Mackowiak, R., Kruse, J., Rother, C., & Köthe, U.: “Training Normalizing Flows with the Information Bottleneck for Competitive Generative Classification”, arXiv:2001.06448 (2020).

Behrmann, J., Duvenaud, D., & Jacobsen, J. H.: “Invertible residual networks”, ICML 2019, arXiv:1811.00995 (2018).

Bellagente, M. et al.: “Invertible Networks or Partons to Detector and Back Again”, arXiv:2006.06685 (2020).

Bengio, Y., et al.: “Representation Learning: A Review and New Perspectives”, IEEE PAMI 35(8):1798-1828 (2013).

Bloem-Reddy, B., & The, Y.W.: “Probabilistic symmetry and invariant neural networks”, JMLR 21(90):1−61(2020).

Chen, R., et al.: “Residual Flows for Invertible Generative Modelling”, NeurIPS (2019).

Dinh, L., Sohl-Dickstein, J., Bengio, S.: “Density estimation using Real NVP”, arXiv:1605.08803 (2016).

Gresele, L., et al.: “Relative gradient optimization of the Jacobian term”, ICML Workshop INNF+, arXiv:2006.15090 (2020).

Gomez, A., et al.: “The Reversible Residual Network: Backpropagation Without Storing Activations”, NIPS (2017).

Gouk, H., et al.: “Regularisation of Neural Networks by Enforcing Lipschitz Continuity”, arXiv:1804.04368 (2018).

Gretton, A., et al.: "A kernel two-sample test.“, JMLR 13:723-773, (2012).

Guo, Ch., et al.: “On calibration of modern neural networks”, ICML (2017).

References

He, K., et al.: “Deep residual learning for image recognition”, CVPR (2016).

Higgins, I., et al.: “beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework”, ICLR (2017).

Khemakhem, I., et al.: “Variational autoencoders and nonlinear ICA: A unifying framework”, AISTAT (2020).

Kingma, D., & Dhariwal, P.: “Glow: Generative flow with invertible 1x1 convolutions”, NIPS (2018).

Kruse, J., et al.: “Benchmarking Invertible Architectures on Inverse Problems”, ICML Workshop INNF (2019).

Ksoll, V. et al.: “Stellar Parameter Determination from Photometry using Invertible Neural Networks”, to appear in: MNRAS, arXiv:2007.08391 (2020).

Lee, W., et al.: “High-Fidelity Synthesis with Disentangled Representation”, arXiv:2001.04296 (2020).

Mackowiak, R., Ardizzone, L. et al.: “Generative Classifiers as a Basis for Trustworthy Computer Vision”, arXiv:2007.15036 (2020).

Putzky, P., & Welling, M.: “Invert to Learn to Invert”, NeurIPS (2019).

Radev, S., et al.: “BayesFlow: Learning Complex Stochastic Models with Invertible Neural Networks”, arXiv:2003.06281 (2020).

Radev, S., et al.: “Model-based Bayesian inference – an application to the COVID-19 pandemics in Germany”, to appear (2020).

Shiono, T.: “Estimation of Agent-Based Models Using Bayesian Deep Learning Approach of BayesFlow”, SSRN:3640351 (2020).

Sorrenson, P., Rother, C., Köthe, U.: “Disentanglement by Nonlinear ICA with General Incompressible-flow Networks (GIN)”, ICLR 2020, arXiv:2001.04872 (2020).

Tishby, N., et al.: “The Information Bottleneck Method”, Allerton Conf. Communications, Control, Computing, arXiv:0004057 (1999).